



**TUGAS AKHIR - TE 141599**

**INTERPRETER BAHASA INDONESIA  
UNTUK PENGATURAN GERAKAN ROBOT NAO**

Eka Prasetyo Herwidodo  
NRP 2210100161

Dosen Pembimbing  
Ahmad Zaini, ST., M.Sc.  
Muhtadin, ST., MT.

JURUSAN TEKNIK ELEKTRO  
Fakultas Teknologi Industri  
Institut Teknologi Sepuluh November  
Surabaya 2015



**ITS**  
Institut  
Teknologi  
Sepuluh Nopember

**FINAL PROJECT - TE 141599**

## **INDONESIAN LANGUAGE INTERPRETER TO CONTROL NAO ROBOT MOVEMENT**

Eka Prasetyo Herwidodo  
NRP 2210100161

Advisor  
Ahmad Zaini, ST., M.Sc.  
Muhtadin, ST., MT.

Departement of Electrical Engineering  
Faculty of Industrial Technology  
Sepuluh Nopember Institute of Technology  
Surabaya 2015

**INTERPRETER BAHASA INDONESIA UNTUK  
PENGATURAN GERAKAN ROBOT NAO**

**TUGAS AKHIR**

**Diajukan untuk Memenuhi Sebagian Persyaratan  
Untuk Memperoleh Gelar Sarjana Teknik  
Pada  
Bidang Studi Teknik Komputer dan Telematika  
Jurusan Teknik Elektro  
Institut Teknologi Sepuluh Nopember**

**Menyetujui:**

**Dosen Pembimbing I**

**Dosen Pembimbing II**

**Ahmad Zaini, ST., M.Sc.**

**NIP. 197504192002121003**

**Muhtadin, ST., MT.**

**NIP. 198106092009121003**



**SURABAYA  
JANUARI, 2015**

# ABSTRAK

Pendidikan di bidang robotika mulai banyak diajarkan pada tingkat pendidikan dasar. Salah satu robot yang digunakan untuk media riset adalah Nao dari Aldebaran Robotics. Robot Nao dilengkapi dengan perangkat *graphical programming Choregraphe* dan *Software development kit (SDK)*. perangkat tersebut berfungsi untuk melakukan pemrograman terhadap robot agar bisa bergerak sesuai kemampuannya. Perangkat yang melengkapi robot Nao ini dirasa belum memberikan kemudahan dalam hal mengenalkan pemrograman robotika terutama bagi Anak SD, dan SMP di Indonesia sehingga diperlukan suatu perangkat/interpreter yang dapat memberikan kemudahan dalam melakukan pemrograman robotika terutama robot humanoid. Interpreter merupakan *software* yang berfungsi sebagai penerjemah bahasa yang dimengerti oleh komputer (bahasa mesin) perintah per perintah. Dalam melakukan fungsinya, interpreter untuk robot Nao terdiri atas 3 bagian yaitu: *Lexer*, *Parser*, dan Eksekutor. Bagian *lexer* membaca masukan perintah dan merubahnya menjadi token dengan menggunakan *Regex*. Bagian parser memproses token menjadi kode representasi tengah sesuai dengan syntax yang dibuat. Bagian eksekutor mengeksekusi kode representasi tengah yang mengirimkan perintah ke robot Nao untuk bergerak. Dari hasil tugas akhir ini dihasilkan sebuah interpreter yang dapat menerima masukan perintah bahasa Indonesia dengan *autocomplete* yang kemudian memprosesnya menjadi gerakan pada robot Nao. Dari 10 anak yang mencoba program Interpreter INI, semuanya mampu membuat gerakan untuk robot NAO.

Kata Kunci: Robot NAO, Pemrograman, Interpreter, Perintah Bahasa Indonesia, Gerakan.

*Halaman ini sengaja dikosongkan*

# ABSTRACT

Education in the field of robotics recently has began to taught at elementary level of education. One of the robots that are used for media of research is Nao from Aldebaran Robotics. Nao robot comes with some tools: Choregraphe and software development kit (SDK). The tools serves to conduct the programming of the robot in order to move accordance with its ability. Tools that complement Nao robot is deemed not provide ease in terms programming its movement especially for elementary school and middle school children in Indonesia. So we need a tool / interpreter that can provide ease of programming particularly in Nao robot. Interpreter is a software that acts as translator to language understood by the computer (machine language) command per command. In conducting its functions, the interpreter for Nao robot consists of three parts: Lexer, Parser, and Executor. Part lexer reads the input command and turn it into a token by using Regular expressions. Part parser process the tokens to become middle code representation according to the syntax that are made. Part executor then execute the code representation which sends commands to Nao robot in order to move. The results of this study produced an interpreter that can receive Indonesian commands as input then process it to perform Nao robot motion. From 10 children that try this INI Interpreter, all of them can make a motion from NAO robot

Keywords: NAO Robot, Programming, Interpreter, Command in Indonesian Bahasa, Movement.

*Halaman ini sengaja dikosongkan*

# KATA PENGANTAR

Puji dan syukur kehadiran Allah SWT atas segala limpahan berkah, rahmat, serta hidayah-Nya, penulis dapat menyelesaikan penelitian ini dengan judul : **Interpreter Bahasa Indonesia untuk Pengaturan Gerakan Robot NAO.**

Penelitian ini disusun dalam rangka pemenuhan bidang riset di Jurusan Teknik Elektro ITS, Bidang Studi Teknik Komputer dan Telematika, serta digunakan sebagai persyaratan menyelesaikan pendidikan S1. Penelitian ini dapat terselesaikan tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Keluarga, Ibu dan Bapak tercinta yang telah memberikan dorongan spiritual dan material dalam penyelesaian buku penelitian ini.
2. Bapak Dr. Tri Arief Sardjono, S.T., M.T. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember.
3. Secara khusus penulis mengucapkan terima kasih yang sebesar-besarnya kepada Bapak Ahmad Zaini, ST., M.Sc. dan Bapak Muhtadin, ST., MT. atas bimbingan selama mengerjakan penelitian.
4. Bapak-ibu dosen pengajar Bidang Studi Teknik Komputer dan Telematika, atas pengajaran, bimbingan, serta perhatian yang diberikan kepada penulis selama ini.
5. Seluruh teman-teman angkatan e-50 serta teman-teman *B201-crew* Laboratorium Bidang Studi Teknik Komputer dan Telematika.

Kesempurnaan hanya milik Allah SWT, untuk itu penulis memohon segenap kritik dan saran yang membangun. Semoga penelitian ini dapat memberikan manfaat bagi kita semua. Amin.

Surabaya, Januari 2015

Penulis



*Halaman ini sengaja dikosongkan*

# DAFTAR ISI

Abstrak	i
Abstract	iii
KATA PENGANTAR	v
DAFTAR ISI	vii
DAFTAR GAMBAR	ix
DAFTAR TABEL	xi
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar belakang . . . . .	1
1.2 Rumusan masalah . . . . .	1
1.3 Tujuan . . . . .	2
1.4 Batasan masalah . . . . .	2
1.5 Sistematika Penulisan . . . . .	2
<b>2 TINJAUAN PUSTAKA</b>	<b>5</b>
2.1 Robot . . . . .	5
2.1.1 Prinsip Kerja Robot . . . . .	6
2.1.2 Derajat Kebebasan Gerak . . . . .	6
2.1.3 Tipe Pergerakan . . . . .	7
2.1.4 Pose, <i>Motion</i> , <i>Behavior</i> . . . . .	7
2.2 Perangkat Keras Nao Aldebaran . . . . .	8
2.3 Perangkat Lunak Nao Aldebaran . . . . .	9
2.3.1 Naoqi . . . . .	10
2.3.2 Choregraphe . . . . .	12
2.3.3 NAOsim . . . . .	14
2.4 <i>Regular Expression</i> . . . . .	15
2.5 Interpreter . . . . .	17
2.5.1 <i>Lexer</i> . . . . .	20
2.5.2 <i>Parser</i> . . . . .	22
2.5.3 <i>Eksekutor</i> . . . . .	22
2.6 <i>Auto Completer</i> . . . . .	23

<b>3</b>	<b>DESAIN DAN IMPLEMENTASI</b>	<b>25</b>
3.1	Desain Sistem . . . . .	25
3.2	Alur Kerja . . . . .	26
3.3	Implementasi <i>Lexer</i> . . . . .	27
3.4	Implementasi <i>Parser</i> . . . . .	28
3.4.1	Perintah tanpa parameter . . . . .	29
3.4.2	Perintah dengan parameter . . . . .	30
3.4.3	Perulangan . . . . .	32
3.4.4	<i>Parallel processing</i> . . . . .	32
3.4.5	Pengaturan kecepatan . . . . .	33
3.4.6	Pengucapan kalimat . . . . .	34
3.5	Eksekutor . . . . .	34
3.5.1	Eksekutor untuk perintah gerakan . . . . .	35
3.5.2	Eksekutor untuk deteksi pose pada robot NAO . . . . .	35
3.5.3	Eksekutor Untuk Pengucapan . . . . .	36
3.6	<i>User Interface</i> . . . . .	37
<b>4</b>	<b>PENGUJIAN DAN ANALISA</b>	<b>41</b>
4.1	Pengujian syntax . . . . .	41
4.1.1	Syntax benar . . . . .	41
4.2	Syntax salah . . . . .	42
4.3	Pengujian keseimbangan robot . . . . .	43
4.3.1	Pengujian kecepatan terhadap keseimbangan . . . . .	44
4.3.2	Pengujian kombinasi perintah terhadap keseimbangan . . . . .	44
4.4	Survei Penggunaan Interpreter . . . . .	57
<b>5</b>	<b>PENUTUP</b>	<b>67</b>
5.1	Kesimpulan . . . . .	67
5.2	Saran . . . . .	67
	<b>DAFTAR PUSTAKA</b>	<b>69</b>

## DAFTAR TABEL

2.1	Spesifikasi Robot [3] . . . . .	8
2.2	Nilai <i>Joint</i> Robot <i>NAO</i> [3] . . . . .	11
2.3	Beberapa formula pada <i>regex</i> . . . . .	16
3.1	List <i>Regex</i> untuk implementasi lexer . . . . .	28
3.2	Perintah tanpa parameter . . . . .	39
3.3	Perintah dengan parameter . . . . .	40
3.4	Batas sudut . . . . .	40
4.1	Gerakan berdiri dengan syntax masukan benar . . .	42
4.2	Gerakan membungkuk dengan syntax masukan benar	43
4.3	Gerakan cium dengan syntax masukan benar . . . .	46
4.4	Gerakan dansa tangan dengan syntax masukan benar	47
4.5	Gerakan duduk dengan syntax masukan benar . . .	48
4.6	Gerakan hormat dengan syntax masukan benar . . .	49
4.7	Gerakan jongkok dengan syntax masukan benar . . .	49
4.8	Gerakan kecapekan dengan syntax masukan benar .	50
4.9	Gerakan hadap kanan dengan syntax benar . . . . .	51
4.10	Gerakan tengok kanan syntax masukan benar . . . .	51
4.11	Kesalahan berupa penulisan syntax . . . . .	52
4.12	Kesalahan berupa penulisan parameter . . . . .	53
4.13	Kesalahan berupa tidak menambahkan tanda (;) pada akhir perintah . . . . .	54
4.14	Gerakan duduk dengan kecepatan standar . . . . .	55
4.15	Gerakan duduk dengan kecepatan 5 . . . . .	55
4.16	Gerakan duduk dengan kecepatan 6 . . . . .	56
4.17	Gerakan duduk dengan kecepatan 7 . . . . .	56
4.18	Gerakan berdiri dengan kecepatan standar . . . . .	57
4.19	Gerakan berdiri dengan kecepatan 5 . . . . .	57
4.20	Gerakan berdiri dengan kecepatan 6 . . . . .	58
4.21	Gerakan berdiri dengan kecepatan 7 . . . . .	58
4.22	Gerakan taichi dengan kecepatan standar . . . . .	59
4.23	Gerakan taichi dengan kecepatan 5 . . . . .	60
4.24	Gerakan taichi dengan kecepatan 6 . . . . .	61
4.25	Gerakan taichi dengan kecepatan 7 . . . . .	62
4.26	Condong ke belakang + angkat lengan kiri . . . . .	62

4.27	Condong ke belakang + angkat lengan kanan . . . .	63
4.28	angkat kaki kanan + angkat lengan kiri . . . . .	64
4.29	angkat kaki kiri + angkat lengan kanan . . . . .	65

# DAFTAR GAMBAR

2.1	Interaksi antar komponen robot [1]	6
2.2	Pose, Motion, dan Behavior[2]	8
2.3	Joint pada NAO versi 3.3[3]	10
2.4	Naoqi Framework[3]	12
2.5	Hubungan broker-libraries-modules [3]	13
2.6	Hubungan <i>broker-modules-method</i> [3]	14
2.7	Tampilan Choregraphe[3]	15
2.8	Tampilan NAOsim[3]	16
2.9	<i>Pure Interpreter</i> [13]	19
2.10	<i>Lexer</i>	20
2.11	<i>Tokenicer</i> [4]	21
2.12	<i>Parser</i>	22
2.13	Tampilan box Auto Completer	24
3.1	<i>Desain Sistem</i>	25
3.2	Alur Kerja	27
3.3	Alur kerja tokenisasi	28
3.4	Implementasi perintah tanpa parameter	30
3.5	Batas Sudut Kepala [3]	31
3.6	Batas Sudut Tangan [3]	32
3.7	Proses pada syntax perulangan	33
3.8	Pose robot NAO[3]	36
3.9	<i>User Interface</i>	38
3.10	Diagram alur <i>Auto Completer</i>	38

*Halaman ini sengaja dikosongkan*

# BAB 1

## PENDAHULUAN

### 1.1 Latar belakang

Sebagai salah satu inovasi teknologi yang sedang berkembang pesat, ilmu robotika saat ini mulai dianggap penting untuk dipelajari. Pendidikan di bidang robotika mulai banyak diajarkan pada tingkat pendidikan dasar. Dengan pengenalan yang lebih awal, diharapkan menimbulkan ketertarikan untuk mempelajari teknologi robotika secara lebih dalam.

Salah satu robot yang digunakan untuk media riset adalah NAO dari Aldebaran Robotics. NAO memiliki beberapa versi derajat kebebasan, antara lain: 14, 21, dan 25 derajat kebebasan. Derajat kebebasan / *Degree of Freedom* (DoF) adalah parameter kebebasan gerakan dari robot. Robot NAO dilengkapi dengan perangkat *graphical programming* Choregraphe dan *Software development kit* (SDK). Perangkat tersebut berfungsi untuk melakukan pemrograman terhadap robot agar bisa bergerak sesuai keinginan, dan sesuai dengan potensi derajat kebebasan yang ada. Perangkat yang melengkapi robot NAO ini sebenarnya sudah relatif mudah dimengerti bagi seseorang yang menguasai dasar-dasar pemrograman robot. Untuk dapat membuat sebuah gerakan pada robot NAO, pengguna harus menentukan jalur interpolasi dari *joint-joint* yang ada pada robot NAO. Selain itu bahasa yang digunakan untuk pemrograman robot NAO adalah bahasa Inggris yang rata-rata masih belum bisa dipahami oleh seorang anak dengan pendidikan SD dan SMP di Indonesia. Oleh karena itu, diperlukan sebuah aplikasi interpreter yang dapat melakukan interpretasi dari perintah bahasa Indonesia untuk mengatur gerakan robot NAO. Interpreter tersebut harus dapat melakukan tugasnya dengan memahami tiap perintah yang dimasukkan dan merubahnya menjadi kode program yang dapat mengatur gerakan robot NAO.

### 1.2 Rumusan masalah

1. Perangkat pemrograman yang disediakan oleh Aldebaran masih belum bisa memberikan kemudahan dalam hal memperkenalkan pemrograman robotika terutama bagi anak SD dan



SMP. Pengguna harus mengatur nilai interpolasi dari *joint-joint* yang ada pada robot NAO untuk membuat gerakan pada robot NAO.

2. Bahasa yang digunakan pada perangkat pemrograman yang disediakan oleh Aldebaran adalah bahasa inggris yang membuat bertambahnya tingkat kesulitan bagi anak usia SD dan SMP di Indonesia.

### 1.3 Tujuan

Tujuan dari Tugas Akhir ini adalah membuat sebuah interpreter yang menerjemahkan perintah bahasa Indonesia untuk mengatur gerakan robot NAO. Interpreter yang dikembangkan ini diharapkan dapat mempermudah pengaturan gerakan robot NAO.

### 1.4 Batasan masalah

Untuk memfokuskan permasalahan yang akan diangkat maka dilakukan pembatasan masalah. Batasan-batasan masalah tersebut diantaranya adalah:

1. Sebagai masukan Interpreter adalah perintah dengan kosakata bahasa Indonesia yang sesuai dengan syntax yang telah dibuat.
2. Perintah yang dimasukan bukan berupa kalimat melainkan berupa perintah sederhana yang tidak kompleks.
3. Robot NAO harus terkoneksi dengan PC dimana Interpreter ini dijalankan, baik secara *wired* ataupun *wireless*.

### 1.5 Sistematika Penulisan

Laporan penelitian tugas akhir ini tersusun dalam sistematika dan terstruktur sehingga lebih mudah dipahami dan dipelajari oleh pembaca maupun seseorang yang hendak melanjutkan penelitian ini. Sistematika laporan penelitian ini berdasarkan alur sebagai berikut.

#### BAB I PENDAHULUAN

Bab ini berisi uraian tentang latar belakang permasalahan, pegasan dan alasan pemilihan judul, tujuan penelitian, metodologi penelitian dan sistematika laporan.

#### BAB II TEORI PENUNJANG

Pada bab ini berisi tentang uraian secara sistematis teori-teori

yang berhubungan dengan permasalahan yang dibahas pada pengerjaan tugas akhir ini. Teori-teori ini digunakan sebagai dasar dalam pengerjaan, yaitu informasi terkait teknologi robot nao Aldebaran, teori mengenai Interpreter, dan teori-teori penunjang lainnya.

### BAB III PERANCANGAN SISTEM APLIKASI

Bab ini berisi tentang penjelasan-penjelasan terkait sistem yang dibuat. Guna mendukung itu, digunakanlah blok diagram agar sistem yang akan dibuat dapat terlihat dan mudah dibaca untuk diimplementasikan pada pembuatan aplikasi perangkat lunak.

### BAB IV PENGUJIAN DAN ANALISA

Bab ini menjelaskan tentang pengujian yang dilakukan terhadap sistem hasil dari tugas akhir ini dan menganalisa sistem tersebut. Spesifikasi perangkat keras dan perangkat lunak yang digunakan juga disebutkan dalam bab ini. Sehingga ketika akan dikembangkan lebih jauh, spesifikasi perlengkapannya bisa dipenuhi tanpa harus melakukan uji coba perangkat lunak maupun perangkat keras lagi.

### BAB V PENUTUP

Bab ini merupakan penutup yang berisi kesimpulan dari sistem perangkat lunak yang telah di buat dari hasil pengujian yang telah dilakukan. Saran dan kritik yang membangun untuk pengembangannya lebih lanjut juga dituliskan pada bab ini.

*Halaman ini sengaja dikosongkan*

## BAB 2

### TINJAUAN PUSTAKA

Pada bab 2, akan dibahas mengenai teori penunjang, perangkat keras, dan perangkat lunak yang digunakan sebagai bahan acuan dan referensi agar tugas akhir ini menjadi lebih terarah

#### 2.1 Robot

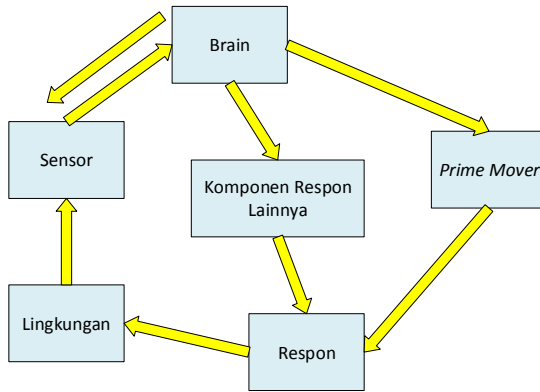
Kata Robot pertama kali muncul pada tahun 1921 dalam sebuah drama berjudul *R.U.R (Rossums Universal Robot)*, karangan Karel Capek. Kata robot berasal dari bahasa ceko *robota* yang berarti *forced labor* [1].

Sebuah robot adalah sebuah unit baik berupa mekanikal atau fisikal maupun virtual yang memiliki kecerdasan. Pada umumnya robot berupa rangkaian elektro mekanik yang dapat bergerak dan memiliki kemampuan berpikir. Namun, hingga saat ini definisi dari sebuah mesin atau alat dapat dikategorikan sebagai robot masih terus diperdebatkan dan dibakukan [1].

Secara umum, sebuah robot memiliki sifat-sifat atau karakteristik sebagai berikut [5] :

1. *Sensing*, Sebuah robot harus bisa mendeteksi atau menerima rangsangan dari lingkungan sekitarnya
2. *Movement*, Sebuah robot harus bisa bergerak baik itu dilakukan dengan menggunakan roda, berjalan dengan kaki atau didorong oleh pendorong
3. *Energy*, Sebuah robot harus memiliki energi yang digunakan untuk dirinya sendiri baik energi listrik, energi matahari, atau dari baterai
4. *Intelligence*, Sebuah robot harus memiliki suatu kecerdasan atau kemampuan berpikir. Kecerdasan tersebut diberikan oleh seorang programmer yang memprogram robot

*International Standart ISO 8373* mendefinisikan robot sebagai :  
”an automatically controlled, reprogrammable, multipurpose, manipulator programmable in three or more axes, which maybe either fixed or mobile for use in industrial automation applications.” Banyak bidang dan aspek yang memanfaatkan robot antara lain: industri, kedokteran, pelayanan, permainan, pendidikan dan lain-lain.



Gambar 2.1: Interaksi antar komponen robot [1]

### 2.1.1 Prinsip Kerja Robot

Sampai saat ini belum ada robot yang mampu berinteraksi secara mandiri atau dengan kata lain bertindak seperti manusia yang dapat melakukan banyak hal tanpa harus menunggu suatu rangsangan dari lingkungan. Secara sekilas robot terlihat hidup, tetapi sebenarnya robot hanya merespon rangsangan yang diterima dari lingkungan sekitarnya. Pada Gambar 2.1 dapat dipahami bahwa robot hanya merespon rangsangan dari lingkungan menggunakan sensor yang ada pada robot, sensor berfungsi sebagai indera pada robot layaknya indera manusia. Kemudian setelah rangsangan tersebut diterima oleh sensor, selanjutnya robot memprosesnya dengan menggunakan pusat pengolahan data dari robot. Setelah itu pusat pengolahan data tersebut (*Brain*) akan mengirim perintah ke komponen respon atau komponen penggerak utama untuk memberi respon ke lingkungan sehingga seolah-olah robot mengerti dan memahami respon dari lingkungan.

### 2.1.2 Derajat Kebebasan Gerak

Derajat kebebasan gerak atau *degree of freedom (DOF)* dari sebuah sistem robot dapat dibandingkan dengan dengan bagaimana tubuh manusia bergerak. Sama halnya dengan manusia, derajat

kebebasan gerak yang dibutuhkan untuk menggerakkan sebuah lengan robot humanoid untuk mencapai fleksibilitas maksimum adalah enam buah [6]. Meski enam derajat kebebasan gerak dibutuhkan untuk mencapai fleksibilitas maksimum, namun kebanyakan robot hanya menggunakan 3 sampai 5 derajat kebebasan gerak. Semakin banyak derajat kebebasan gerak semakin kompleks pergerakan yang dapat dilakukan dan semakin kompleks juga pemrogramannya. Tiga jenis derajat kebebasan gerak pada robot yaitu: *Pitch*, *Yaw* dan *Roll* [7].

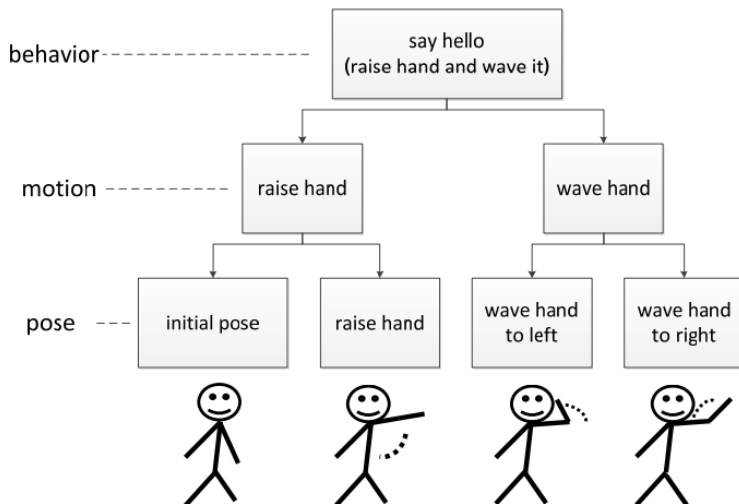
### 2.1.3 Tipe Pergerakan

Berdasarkan tipe pergerakannya, robot dapat dibagi menjadi empat jenis [8] yaitu:

1. *Slow motion*, gerakan dari satu titik ke titik yang lain dengan kecepatan tetap
2. *Joint-interpolated motion*, membutuhkan kendali untuk mengkalkulasi setiap gerak dengan kecepatan yang variatif
3. *Straight-Line Interpolation Motion*, gerak lurus.
4. *Circular Interpolation Motion*, gerak memutar.

### 2.1.4 Pose, *Motion*, *Behavior*

Pose berarti momen diam dari seluruh posisi tubuh robot. Mungkin definisi dari pose sudah jelas dilihat dari istilah yang digunakan. Namun untuk definisi antara *motion* dan *behavior* agak abu-abu sehingga perlu untuk dijelaskan lebih dalam. Berdasarkan Thesis dari Min-Chang Wu didefinisikan bahwa *motion* terdiri atas setidaknya 2 pose yang berbeda. Sedangkan *behavior* setidaknya terdiri atas 2 *motion*[2]. Definisi dari pose, *motion*, dan *behavior* secara lebih jelas diilustrasikan pada Gambar 2.2. Pada gambar dapat diamati hirarki dari *behavior*, *motion*, dan *pose*. Pada tingkat tertinggi adalah *behavior* "say hello" dimana *behavior* ini adalah sebuah *behavior* yang disusun dari 2 *motion* yaitu *motion* "raising hand" dan *motion* "waving hand". *Motion* "raising hand" adalah *motion* yang terdiri dari 2 pose yaitu pose "initial" dan pose "raising hand" lalu *motion* "waving hand" juga terdiri dari 2 pose yaitu "waving hand to left" dan "waving hand to right".



Gambar 2.2: Pose, Motion, dan Behavior[2]

## 2.2 Perangkat Keras Nao Aldebaran

Robot humanoid NAO dikembangkan oleh *Aldebaran robotics* sejak 2005, robot ini dikembangkan dengan 25 macam kemampuan gerak atau biasa disebut *degree of freedom (DOF)*. Robot humanoid ini memiliki tinggi 57 cm dengan bobot 4.5 kg. Bobot ini sudah termasuk baterai yang digunakan pada robot. Lama daya tahan baterai bila kondisi *autonomous* dapat digunakan hingga 90 menit, hal ini tergantung dari perilaku yang dilakukan oleh robot [3]. Spesifikasi dari Robot humanoid nao dapat dilihat pada Tabel 2.1.

Tabel 2.1: Spesifikasi Robot [3]

<i>Operating system</i>	<i>Linux platform</i>
<i>Processor</i>	AMD Geode
<i>Speed Clock</i>	500 MHz
RAM	256 MB SDRAM
<i>Flash memory</i>	1 GB

Ada dua cara komunikasi yang digunakan antara modul *external* (robot humanoid nao) dan modul *internal* (aplikasi), yaitu dengan Ethernet atau Wi-Fi. *Aldebaran robotics* telah mengeluarkan beberapa versi untuk robot humanoid nao Aldebaran antara lain:

1. Edisi *Robocup* dirilis dengan 21 DOF pada Maret 2008.
2. Edisi *Academics* dirilis dengan 25 DOF pada tahun 2008.
3. NAO H25 Versi 3.3 dirilis dengan 25 DOF, memiliki lengan yang lebih panjang, dan memiliki beberapa versi kepala pada tahun 2010.

Adapun robot NAO yang digunakan dalam tugas akhir ini adalah NAO H25 versi 3.3. Robot humanoid NAO H25 versi 3.3 memiliki beberapa sensor yang dapat digunakan, sensor itu antara lain:

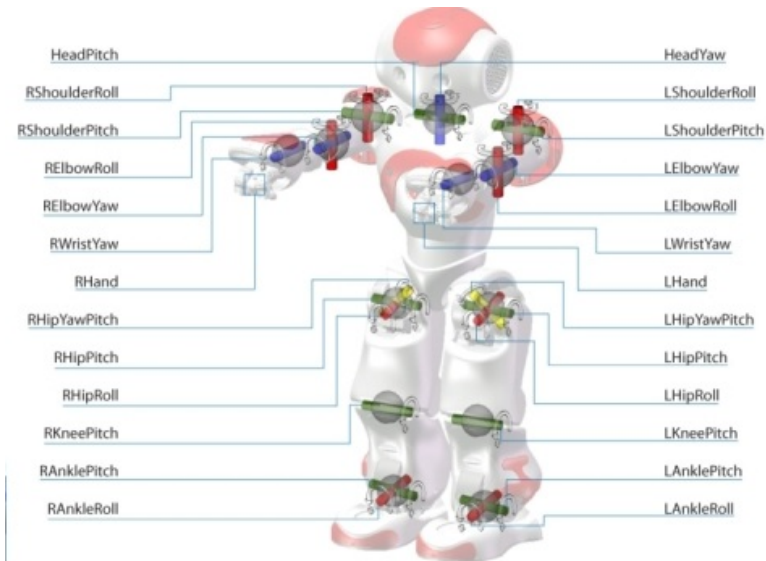
1. Dua *gyrometer* dan sebuah *accelerometer* yang digunakan untuk melakukan estimasi posisi dan orientasi dari robot. Sensor ini terletak di dalam tubuh robot.
2. Dua kamera yang terletak di kepala.
3. Empat *microphone* yang terletak di sekitar kepala.
4. Dua *ultrasonic transmitter* dan *receiver* terletak di depan bagian dada.
5. Empat *force sensitive resistor* yang terletak pada masing-masing pergelangan kaki.
6. Sensor sentuh yang terletak pada kepala dan lengan.

Aplikasi yang dibuat pada tugas akhir ini melibatkan komponen mekanik pada robot (*joint robot*). Untuk itu diperlukan gambaran mengenai nilai maksimum dan minimum untuk *joint-joint* yang digunakan dalam aplikasi ini. Gambar 2.3 menunjukkan joint yang terdapat pada NAO H25 versi 3.3. Tabel 2.2 menunjukkan informasi mengenai nilai maksimum dan minimum untuk *joint-joint* yang digunakan dalam aplikasi ini.

## 2.3 Perangkat Lunak Nao Aldebaran

Sebelumnya telah dijelaskan tentang perangkat keras dari robot humanoid Nao. Selain perangkat keras, robot humanoid Nao juga memiliki perangkat lunak sebagai pendukung. Perangkat Lunak tersebut terdiri dari beberapa elemen pendukung yang akan dijelaskan melalui sub berikut ini.





Gambar 2.3: Joint pada NAO versi 3.3[3]

### 2.3.1 Naoqi

*Naoqi* adalah sebuah aplikasi yang berjalan pada robot untuk mengontrol komponen-komponen yang ada pada robot humanoid Nao [9]. Pada tugas akhir ini *naoqi* digunakan sebagai penghubung antara interpreter yang dibuat dengan aktuator yang ada pada robot humanoid NAO. *Naoqi* dapat melakukan fungsi-fungsi seperti: Parallelisme, sinkronisasi, mengatasi resource, dan mengatasi event. Pada Gambar 2.4 dapat dilihat *framework* dari *Naoqi*. Bagian *naoqi* yang menghubungkan modul-modul yang ada di NAO dengan aplikasi yang ada di luar robot NAO disebut dengan *broker*. *Broker* adalah bagian dari *naoqi* yang bertugas melakukan *listing* seluruh komponen pada *naoqi* kemudian memberikannya pada aplikasi yang memanggil komponen tersebut. Fungsi dari *broker* adalah:

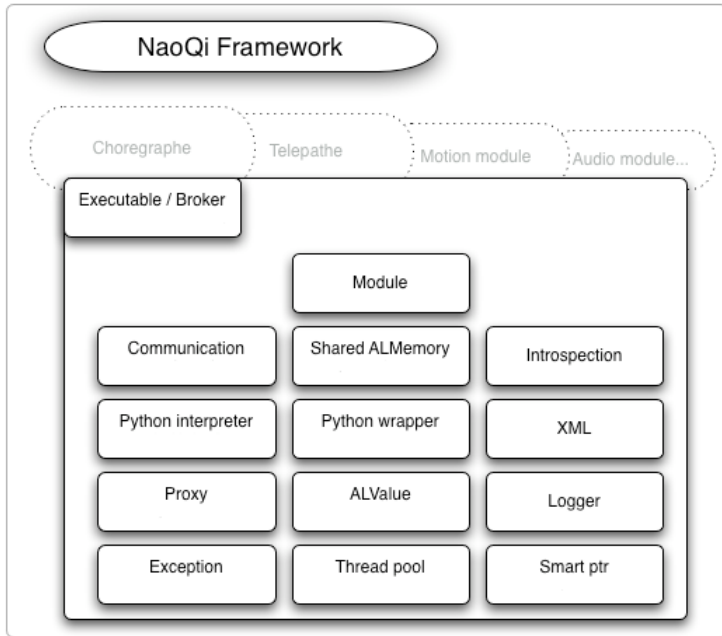
1. *Directory service*, memungkinkan user menemukan modul dan metode yang dimilikinya
2. *Network service*, memungkinkan metode yang ada pada modul

Tabel 2.2: Nilai *Joint Robot NAO* [3]

<i>Joint</i>	<i>Angle Name</i>	<i>Min Value</i>	<i>Max Value</i>
<i>Head</i>	<i>Yaw</i>	-119.5	119.5
	<i>Pitch</i>	-38.5	39.5
<i>RShoulder</i>	<i>Pitch</i>	-119.5	119.5
	<i>Roll</i>	-94.5	-0.5
<i>RElbow</i>	<i>Yaw</i>	-119.5	119.5
	<i>Roll</i>	0.5	89.5
<i>RWrist</i>	<i>Yaw</i>	-104.5	104.5
<i>RHand</i>	<i>Open and Close</i>		
<i>LShoulder</i>	<i>Pitch</i>	-119.5	119.5
	<i>Roll</i>	0.5	94.5
<i>LElbow</i>	<i>Yaw</i>	-119.5	119.5
	<i>Roll</i>	-89.5	-0.5
<i>LWrist</i>	<i>Yaw</i>	-104.5	104.5
<i>LHand</i>	<i>Open and Close</i>		

dapat di akses dari luar proses brokernya.

Modul adalah bagian dari *Naoqi* yang menyimpan *method-method* yang digunakan untuk menjalankan sebuah fungsi tertentu contoh: modul *AlMotion* menyimpan *method-method* yang berhubungan dengan pergerakan robot NAO, lalu modul *AlRobotPose* menyimpan *method-method* yang berhubungan dengan pendeteksian pose pada robot NAO. Pada robot NAO modul disimpan dalam bentuk *library*. *Method* adalah fungsi yang menjalankan kegunaan tertentu yang dapat dipanggil melalui modul. Ketika *Naoqi* dijalankan, broker melakukan proses *load* sebuah file bernama *autoload.ini* yang mendefinisikan semua *library* yang ada pada robot NAO. Kemudian dilakukan *load* untuk *library* tersebut. Setiap *library* memiliki satu atau lebih modul yang kemudian akan menggunakan broker sebagai alat untuk mengumumkan metode-metode yang dimilikinya. Agar lebih jelas hubungan antara *broker*, *library*, dan modul dapat dilihat di Gambar 2.5. Pada gambar dapat dilihat bahwa modul-modul disimpan dalam *library*. Kemudian *library* tersebut di *load* oleh *broker* berdasarkan file *autoload.ini* saat *naoqi* di jalankan.

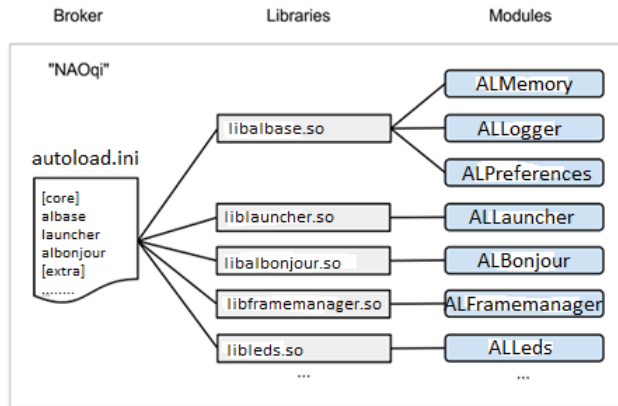


Gambar 2.4: Naoqi Framework[3]

Di dalam setiap modul, terdapat satu atau lebih *method* yang bisa dipanggil untuk dapat menjalankan fungsi tertentu. Pada bagian ini, *broker* adalah bagian yang berperan untuk melakukan listing untuk dapat menemukan method-method ini. Ketika sebuah *library* telah selesai di *load*, modul-modul pada *library* menjadi dapat digunakan method-method yang dimilikinya melalui *broker*. *Broker* juga berperan sebagai perantara bagi method-method ini agar dapat diakses dari luar proses *broker*. Hubungan antara *broker-module-method* dapat dilihat di Gambar 2.6

### 2.3.2 Choregraphe

Choregraphe adalah sebuah aplikasi yang disediakan oleh *Aldebaran Robotics* untuk membuat *behavior* baru melalui box-box yang disediakan di user interface ataupun melalui box-box baru yang di-

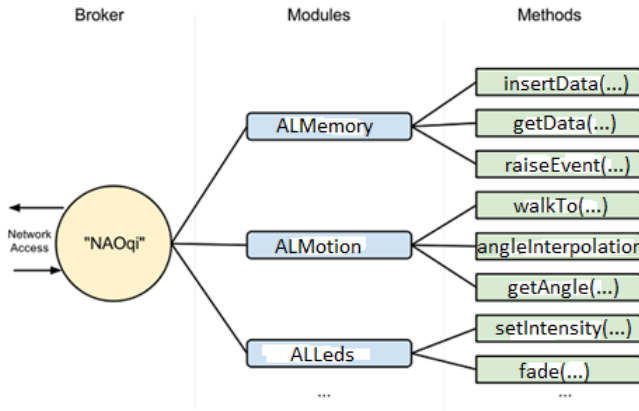


Gambar 2.5: Hubungan broker-libraries-modules [3]

buat oleh user dan kemudian meletakkannya pada halaman proyek sesuai dengan alur yang diinginkan oleh user. Pada choregraphe terdapat tiga tipe box yang terdiri atas:

1. Box *Script*, Tipe box yang didalamnya terdapat skrip python yang dieksekusi saat dilewati alur kerja.
2. Box *Timeline*, Tipe box yang didalamnya terdiri atas pose-pose yang terletak di sebuah garis waktu. Saat alur kerja melewati box tipe ini, maka robot NAO akan merubah pose dirinya sesuai dengan garis waktu tempat pose tersebut.
3. Box *Flow Diagram*, Tipe box yang didalamnya terdapat alur kerja. Tipe box ini digunakan untuk simplifikasi halaman proyek.

Setelah diagram alur yang tersusun atas box-box tersebut telah selesai disusun dan membentuk sebuah *behavior*, hasilnya dapat langsung di jalankan pada komputer yang kemudian akan mengirim perintah ke Naoqi yang ada pada robot atau menyimpannya pada robot untuk kemudian dapat dipanggil setelahnya. Pada choregraphe terdapat visualisasi dari robot NAO yang merepresentasikan kondisi robot NAO nyata pada saat dikoneksikan dengan choregraphe. Choregraphe juga dapat mengirimkan hasil behavior yang telah dibuat

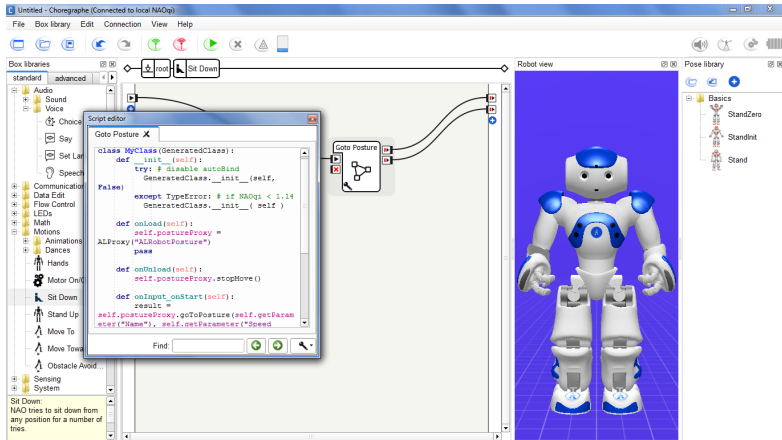


Gambar 2.6: Hubungan *broker-modules-method* [3]

tersebut ke robot NAO virtual dan robot NAO nyata [9]. Tampilan choregraphe dapat dilihat pada Gambar 2.7. Pada Gambar 2.7 dapat dilihat sebuah box pada choregraphe dengan tipe script dibuat dengan menggunakan bahasa pemrograman python.

### 2.3.3 NAOsim

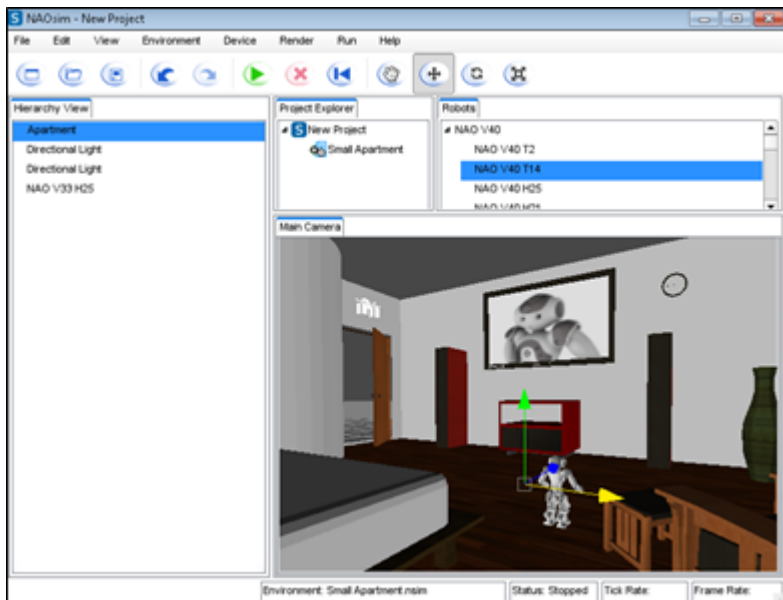
NaoSim adalah sebuah fisik dasar simulasi yang disediakan oleh aldebaran. Tampilan pada NAOsim dapat dilihat pada Gambar 2.8. Melalui gambar tersebut dapat dilihat bahwa pada NaoSim terdapat beberapa fitur yang diberikan yaitu jenis robot yang bervariasi, lingkungan yang dapat diatur, pencahayaan yang dapat disesuaikan dengan kebutuhan [9] dan posisi awal robot yang dapat ditentukan. Pada NaoSim juga terdapat pilihan tipe robot dan juga pilihan versi Naoqi yang akan digunakan sehingga dapat disesuaikan dengan robot yang dimiliki oleh penggunanya. Sama seperti robot NAO yang nyata, robot NAO pada NAOsim juga dapat diprogram dengan perangkat yang sama pada robot asli. Dengan menggunakan *virtual robot* pada NAOsim, tes terhadap gerakan yang dibuat menjadi lebih leluasa tanpa takut terjadi kesalahan yang tidak diinginkan yang menyebabkan kerusakan pada robot.



Gambar 2.7: Tampilan Choregraphe[3]

## 2.4 Regular Expression

*Regular expression* atau *Regex* adalah sebuah formula untuk pencarian pola suatu kalimat/string [10]. *Regular expression* umumnya digunakan oleh banyak pengolah kata/*text editor* dan peralatan lainnya untuk mencari dan memanipulasi kalimat berdasarkan suatu formula tertentu yang merepresentasikan bagian tertentu dari kalimat tersebut. Banyak bahasa pemrograman yang mendukung *regular expression* seperti misalnya PHP, perl, VB dan Tcl. *Regex* dapat digunakan mulai pada tingkat penggunaan terendah untuk mencari sebuah penggalan kata hingga pada tingkat penggunaan tertinggi *regex* mampu melakukan kontrol terhadap data baik mencari, menghapus dan merubah isi data. Dengan menggunakan *regex*, dapat dilakukan perubahan terhadap suatu bagian dari file secara lebih efektif dan lebih mudah. Formula umum yang digunakan pada *regex* dapat dilihat pada tabel 2.3. Dengan menggunakan *regex* pengelompokan suatu kalimat berdasarkan kecocokannya pada *regex* dapat dilakukan. Pada sebuah interpreter *regex* digunakan untuk membuat mekanisme proses tokenisasi yang akan dijelaskan selanjutnya pada bab 2.5.1.



Gambar 2.8: Tampilan NAOsim[3]

Tabel 2.3: Beberapa formula pada *regex*

Pola	Penjelasan
[ ]	Ekspresi kurung digunakan untuk mencocokkan satu karakter yang berada dalam kurung, misal: pola "a[bcd]i" cocok dengan string "abi", "aci", dan "adi". Penggunaan <i>range</i> huruf dalam kurung diperbolehkan, misal : pola "[a-z]" cocok dengan salah satu karakter diantara string "a" sampai "z". Pola [0-9] cocok dengan salah satu angka dari angka nol sampai sembilan. Jika ingin mencari "-" juga, karakter tersebut harus diletakkan di depan atau di belakang kelompok, misal: "[abc-]".

?	Cocok dengan nol atau satu karakter setelahnya. Misal: pola "die?" cocok dengan string "die" dan "di-ed".
+	Cocok dengan satu atau lebih karakter setelahnya. Misal: "yu+k" cocok dengan "yuk", "yuuk", "yuuuk", dan seterusnya.
*	Cocok dengan nol atau lebih karakter sebelumnya. misal: pola "hu*p" cocok dengan string "hp", "hup", "huup" dan seterusnya.
[ ^ ]	Cocok dengan sebuah karakter yang tidak ada dalam kurung, berlawanan dengan yang diatas. misal: pola "[^ abc]" cocok dengan satu karakter apa saja kecuali "a", "b", "c".
{x}	Cocok dengan karakter sebelumnya sejumlah x karakter. misal: pola "[0-9] { }" cocok dengan bilangan berapa saja yang berukuran 3 digit.
{x,y}	Cocok dengan karakter sebelumnya sejumlah x hingga y karakter. misal: pola "[a-z]{3,5}" cocok dengan semua susunan huruf kecil yang terdiri dari 3 sampai 5 huruf.
!	Jika diletakkan di depan pola, maka berarti "bukan". misal pola "!a.u" cocok dengan string apa saja kecuali string "alu", "anu", "abu", "asu", "aiu", dan seterusnya.
^	Jika diletakkan di depan pola, maka proses pencocokan dilakukan hanya pada awal sebuah string. Misal: "^abc" akan cocok dengan "abc" pada "abcde" tetapi tidak pada "babc"

## 2.5 Interpreter

Interpreter adalah program komputer yang mengeksekusi secara langsung instruksi yang dimasukkan tanpa melakukan proses kompilasi[11]. Sebuah interpreter memiliki beberapa metode yang biasanya digunakan untuk melakukan proses interpretasi input[12]. Antara lain:



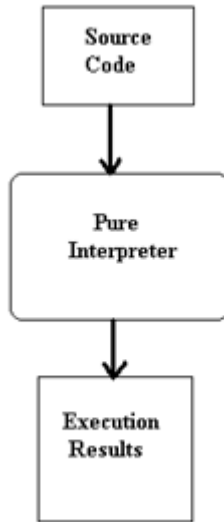
1. melakukan parsing source code dan melakukan tugasnya secara langsung.
2. merubah *source code* menjadi representasi tingkat menengah dan mengeksekusinya.
3. mengeksekusi kode yang sudah dikompil sebelumnya secara eksplisit sesuai dengan *source code*.

Sebuah interpreter memiliki perbedaan dengan kompiler dalam proses eksekusinya di dalam mesin. Sebuah kompiler tidak membutuhkan kode program asal saat proses eksekusi di dalam mesin, karena seluruh *source code* sudah diubah menjadi program yang langsung menjalankan sebuah fungsi tertentu sesuai dengan *source code*. Namun pada interpreter, *source code* masih dibutuhkan karena proses menjalankan mesin dilakukan dengan menerjemahkan perintah baris per baris yang ada pada *source code*, lalu menjalankan kode mesin yang sesuai dengan baris *source code* tersebut. Hal ini membuat interpreter menjadi lebih fleksibel daripada kompiler untuk menjalankan program-program kecil. Hal ini dikarenakan, pengguna tidak perlu melakukan kompilasi ulang sebuah *source code* untuk setiap perubahan perubahan kecil yang dilakukan.

Dapat dilihat pada Gambar 2.9 bahwa sebuah interpreter tidak menghasilkan sebuah file yang dapat dieksekusi. Jadi untuk menjalankan mesin dengan interpreter, hanya diperlukan *source code* yang kemudian dijalankan oleh interpreter. Interpreter dapat dikategorikan menjadi 3 jenis yaitu:

1. *Pure interpreter*
2. *Intermediate-language interpreter*
3. *Execute-type interpreter*

*Pure interpreter* adalah interpreter yang langsung mengeksekusi kode program satu-persatu tanpa melalui *intermediate language*. *Pure interpreter* ini tidak dapat melakukan optimasi sama sekali. Interpreter tipe ini juga tidak dapat melakukan cek Sintak seperti pada kompiler. Contoh dari interpreter tipe ini adalah *Batch script*, *shell script*, dan *command file*. Contohnya pada bat, saat kita menjalankan satu *file script* pada bat, maka bat akan mengeksekusi baris per baris perintah yang ada di bat file tersebut. Jika perintah sukses dieksekusi, maka hasil yang didapatkan adalah hasil yang diinginkan. Namun jika gagal, akan keluar notifikasi kesalahan tanpa diketa-



Gambar 2.9: *Pure Interpreter* [13]

hui kesalahan syntax-nya. *Intermediate-language interpreter* adalah sebuah interpreter yang melakukan penerjemahan source code ke dalam sebuah representasi tengah yang efisien dan mengeksekusinya. Contoh dari Interpreter tipe ini adalah Perl, Matlab, Python, dan Ruby. *Execute-type* interpreter adalah sebuah interpreter yang mengeksekusi code yang sudah dikompilasi berdasarkan masukan yang ada pada *source code*. Contoh dari Interpreter tipe ini adalah UCSD Pascal.

Sedangkan kompiler memiliki cara kerja yang berbeda dari interpreter. Pada kompiler *source code* yang telah dibaca, lalu diterjemahkan sesuai sintak yang berlaku pada kompiler tersebut dan dibuat objeknya. Setelah objek dibuat, dibuatlah penghubung yang menghubungkan seluruh objek, exe, dan seluruh *library* yang dibutuhkan oleh program yang dibuat. Saat pengguna ingin menjalankan program, maka yang dijalankan adalah exe dari program tersebut dan bukan *source code* yang dibuat.

Interpreter dapat dipecah menjadi 3 bagian[13] yaitu:



Gambar 2.10: *Lexer*

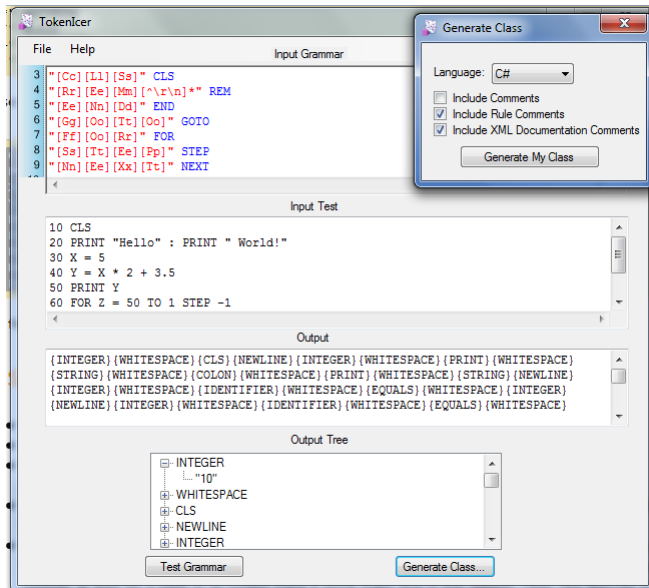
1. Bagian *Lexer*, membagi-bagi masukan perintah menjadi token
2. Bagian *Parser*, memproses token dan merubahnya menjadi representasi tengah yang siap dieksekusi
3. Bagian *Executor*, mengeksekusi representasi tengah

Penjelasan mengenai bagian-bagian tersebut akan dijelaskan pada sub bab berikut.

### 2.5.1 *Lexer*

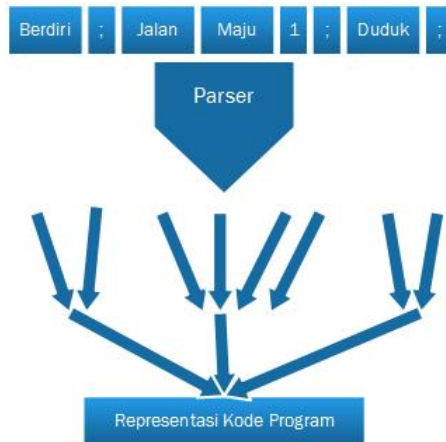
Lexer adalah bagian dari interpreter yang berfungsi melakukan proses *lexing* atau tokenisasi. Dalam proses ini masukan dipecah-pecah menjadi sebuah representasi yang memiliki nilai-nilai seperti integer, indentifier, string, dan lain-lain untuk kemudian disimpan. Representasi dari nilai-nilai tersebut dinamakan token. Secara lebih jelas proses ini dapat diamati pada Gambar 2.10. Tujuan dari proses ini adalah memudahkan pembacaan input perintah pada proses selanjutnya, karena nantinya pembacaan input dilakukan secara bertahap setiap satu token.

Dalam proses tokenisasi diperlukan suatu aturan untuk menetapkan masukan perintah mana yang dirubah menjadi token dan yang tidak. Salah satu perangkat yang digunakan untuk membuat aturan pada proses tokenisasi adalah TokenIcer[4]. TokenIcer adalah sebuah perangkat yang di publikasikan oleh akun icemanind pada website codeproject.com yang mendukung C# pada pemrograman .Net. TokenIcer menerima input berupa *grammar Regex* yang akan kita jadikan aturan dasar proses *lexing*. *Regular Expression* atau yang lebih sering disebut *Regex* merupakan sebuah teknik yang digunakan untuk mencocokkan string teks, seperti ka-



Gambar 2.11: *Tokenicer*[4]

rakter tertentu, kata-kata, atau pola karakter. *Regex* ini digunakan oleh *lexer* sebagai basis dari prosesnya dalam membaca masukan perintah untuk kemudian dipisah menjadi token. Pembuatan kelas *lexer* oleh *TokenIcer* dan tes pemisahan input perintah dapat saya contohkan pada Gambar 2.11 pada kolom teks "input test" dan "output". Dapat diamati pada Gambar 2.11 bahwa masukan yang berupa string dipecah-pecah dan disimpan sesuai dengan masukan *grammar* yang dimasukkan pada kolom "input grammar". Token yang disimpan tadi dapat diambil dan dilihat nilainya secara berurutan sesuai dengan urutan yang dimiliki saat masih berupa string masukan perintah. Setelah masukan *grammar* yang diinginkan dimasukkan dan setelah dilakukan tes masukan dan luarannya, *TokenIcer* ini kemudian dapat membuat sebuah kelas yang dapat digunakan pada pemrograman .Net C#. Proses memasukan *grammar* dan proses menghasilkan luaran berupa kelas yang terjadi pada *TokenIcer* dapat dilihat pada Gambar 2.11.



Gambar 2.12: *Parser*

### 2.5.2 *Parser*

Proses parsing adalah proses mengorganisir seluruh token menjadi sebuah representasi tengah agar dapat dibaca oleh eksekutor. Setelah input dirubah menjadi token dan disimpan, token-token tersebut dipanggil satu persatu sesuai urutan lalu dibaca nilainya untuk kemudian diterjemahkan menjadi kode program yang merepresentasikan input perintah. Proses parsing akan menghasilkan luaran berupa representasi tengah dari masukan perintah yang sudah menjadi token tadi. Namun dalam pembacaan token, jika ditemukan token yang tidak terdapat didalam aturan parser, maka parser juga akan menanganinya baik itu dengan mengeluarkan notifikasi kesalahan atau menghentikan proses parsing. Proses parsing dapat diamati pada Gambar 2.12 . Pada gambar dapat diamati bahwa setiap token dibaca dan dirubah menjadi kode program representasinya.

### 2.5.3 *Eksekutor*

Eksekutor adalah bagian dari interpreter yang mengeksekusi representasi tengah dari parser. Eksekutor merupakan bagian yang paling mudah untuk dibuat karena tugas dari eksekutor hanyalah

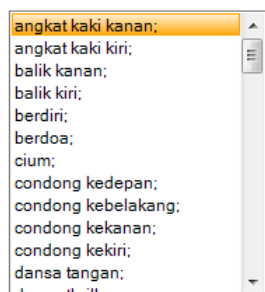
mengeksekusi representasi tengah saja. Pada interpreter yang dibuat untuk komputer, eksekutor yang digunakan adalah mekanisme untuk memberi perintah kepada komponen-komponen yang ada pada komputer. Sedangkan pada robot NAO, eksekutor dapat dibuat dengan Naoqi *Software Development Kit* yang disediakan oleh Aldebaran. Naoqi *SDK* merupakan platform pemrograman yang dibuat untuk mengakses Naoqi. Aldebaran robotics menyediakan support berbagai bahasa untuk SDK ini, diantaranya yang paling umum digunakan adalah: python, c++, urby, dan .Net . Dengan *SDK* ini, dapat dibuat program yang dapat memberikan perintah kepada Naoqi yang berjalan pada robot untuk menggerakkan aktuatornya.

## 2.6 *Auto Completer*

*Auto Completer* adalah sebuah fitur yang memungkinkan pengguna untuk memilih pilihan kata yang sesuai dengan kata yang diperkirakan ingin di masukkan oleh pengguna. Salah satu *toolbox* yang digunakan untuk mengimplementasikan *Auto completer* adalah *Auto Completer* buatan Pavel Torgashov yang di publikasikan pada website codeproject.com [15]. *Auto completer* ini akan menampilkan sebuah box panel, dimana didalamnya terdapat pilihan kata yang dapat melengkapi karakter yang sebelumnya diinputkan oleh pengguna. *Auto completer* buatan Pavel Torgashov ini sudah dibuat menjadi sebuah toolbox yang dapat diimport pada proyek visual studio. Menu penting yang terdapat pada toolbox ini antara lain:

1. *Menu Items*: Pada menu ini dimasukan daftar string dari kata yang ingin dilengkapi.
2. *Appear Interval*: Pada menu ini dapat di atur interval dari munculnya panel *auto completer* dengan masukan oleh pengguna.
3. *MinFragmentLength*: Pada menu ini dapat di atur jumlah karakter per-kata minimum yang dimasukan sebelum ditampilkan panel *auto completer*.

Agar lebih jelas, contoh tampilan menu box *Auto completer* pada sebuah program dapat diamati pada Gambar 2.13



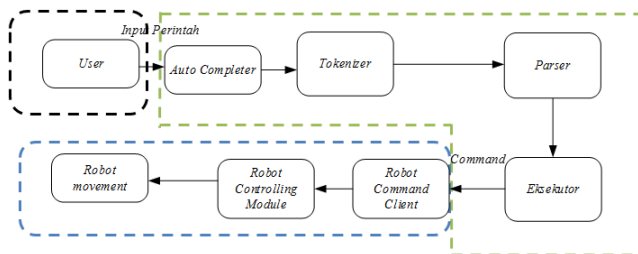
Gambar 2.13: Tampilan box Auto Completer

## BAB 3

# DESAIN DAN IMPLEMENTASI

### 3.1 Desain Sistem

Proses yang terjadi pada interpreter: Indonesian Nao Interpreter (INI) dimulai dengan proses memasukan perintah melalui interface dari interpreter INI. masukan perintah yang dimasukkan pada interface adalah berupa perintah berbahasa Indonesia yang sesuai dengan syntax yang sudah ditetapkan agar masukan perintah dapat di interpretasikan. Dalam proses memasukkan perintah, pada interface juga digunakan *autocomplete* untuk memudahkan dan menghindari kesalahan pada proses memasukan perintah. Kesalahan pada proses memasukan perintah yang biasanya terjadi adalah kesalahan pemasukan perintah yang tidak sesuai syntax. Setelah proses memasukkan perintah, proses selanjutnya adalah merubah masukan perintah menjadi token. Token pada interpreter adalah representasi perintah setelah dipecah-pecah menjadi unit-unit yang memiliki nilai untuk diproses seperti indentifier, keyword, operator, dan lain-lain. Proses merubah masukan perintah menjadi token ini



Keterangan :

- Bagian User
- Bagian Perangkat lunak (Interpreter)
- Bagian Perangkat keras (Robot NAO)

Gambar 3.1: *Desain Sistem*

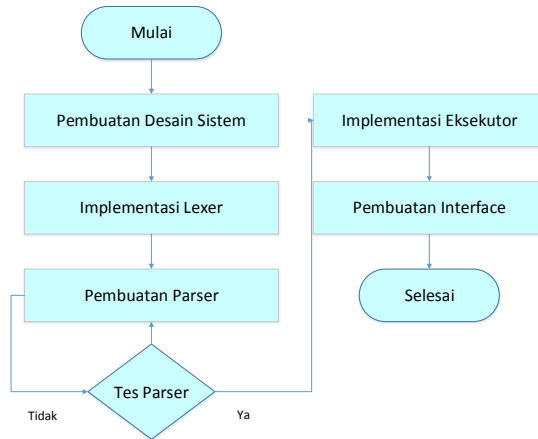


dilakukan oleh sebuah lexer atau *tokenizer* sedangkan untuk prosesnya dinamakan *lexing* atau tokenisasi. Lexer akan menghiraukan *whitespace* karena tidak akan dilakukan interpretasi pada karakter tersebut. Proses selanjutnya setelah melakukan tokenisasi adalah proses parsing yang dilakukan oleh parser. Parser dibutuhkan untuk mengorganisir token menjadi sebuah representasi tengah agar dapat dimengerti dan dijalankan oleh eksekutor. Secara sederhana, parser bertanggung jawab menerjemahkan token menjadi kode program yang siap dieksekusi. Tanpa adanya parser, program tidak dapat melanjutkan ke proses selanjutnya yaitu proses eksekusi. Proses selanjutnya adalah proses eksekusi yang dilakukan oleh eksekutor. Eksekutor adalah bagian dari interpreter yang melakukan proses eksekusi. Proses eksekusi sendiri merupakan proses termudah dari sebuah interpreter. Pada proses eksekusi, program mengirimkan perintah kepada robot NAO untuk menjalankan gerakan sesuai dengan hasil proses parsing dari parser. Perintah dari program tersebut diterima oleh Naoqi yang berjalan pada robot NAO ataupun pada sebuah simulator. Kemudian Naoqi menggerakkan aktuator yang ada pada robot NAO sesuai dengan perintah dari program. Desain sistem pada program ini dapat diilustrasikan pada Gambar 3.1

## 3.2 Alur Kerja

Alur kerja dalam pengerjaan tugas akhir ini terbagi oleh enam tahapan proses. Dalam masing-masing proses memiliki luaran yang dihasilkan dan menjadi inputan dari proses selanjutnya. Tahapan proses dari tugas akhir ini adalah sebagai berikut:

1. Pembuatan desain sistem untuk interpreter.
2. Implementasi *lexer* pada interpreter. Implementasi lexer dilakukan dengan menggunakan lexer *TokenIcer* yaitu lexer yang dibuat dan dipublikasikan oleh akun icemanind pada [codeproject.com](https://codeproject.com)
3. Pembuatan *Parser* pada interpreter, yaitu membuat parser sesuai dengan syntax yang ditetapkan. Daftar syntax dapat dilihat pada halaman lampiran.
4. Tes Parser pada interpreter, dilakukan tes terhadap parser yang sudah di buat.
5. Implementasi eksekutor. Dilakukan implementasi eksekutor



Gambar 3.2: Alur Kerja

untuk control robot NAO sesuai dengan syntax yang telah dibuat.

6. Pembuatan Interface. Disini penulis membuat interface dari interpreter berbahasa indonesia ini.

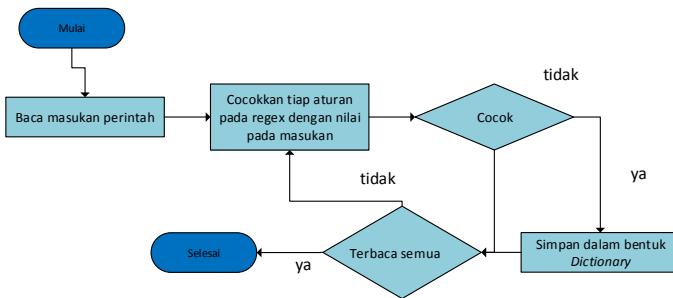
Keseluruhan alur kerja dari tugas akhir ini dapat dilihat pada Gambar 3.2

### 3.3 Implementasi *Lexer*

Bagian interpreter yang pertama kali dibuat adalah *lexer*. Seperti yang dikatakan sebelumnya, *lexer* adalah bagian yang berfungsi melakukan proses lexing atau tokenisasi. Implementasi *Lexer* pada interpreter INI dilakukan dengan membuat aturan *regex* untuk kemudian dimasukan ke TokenIcer. TokenIcer kemudian membuat kelas yang bisa dipanggil dan berfungsi untuk memecah masukan perintah berdasarkan tipe masukannya berdasarkan aturan regex yang telah di masukan ke dalam TokenIcer. Pada interpreter INI pemecahan input dilakukan dengan menggunakan *regex* seperti yang dapat dilihat pada Tabel 3.1. Diagram alur dari proses tokenisasi yang dilakukan oleh kelas yang dibuat oleh TokenIcer dapat dilihat pada Gambar 3.3.

Tabel 3.1: List *Regex* untuk implementasi lexer

RegEx	Keterangan
[Ee][Nn][Dd]	Untuk mendeteksi perintah end
\".*\?"	Untuk mendeteksi string pada input perintah
[a-zA-Z_][a-zA-Z0-9_]*	Untuk mendeteksi syntax perintah pada input
[0-9]?\\.\+[0-9]+	Untuk mendeteksi float pada input
[\\+\\-0-9]+	Untuk mendeteksi interger pada input
\(	Untuk mendeteksi tanda ( pada input
\)	Untuk mendeteksi tanda ) pada input
\}	Untuk mendeteksi tanda } pada input
\{	Untuk mendeteksi tanda { pada input
;	Untuk mendeteksi tanda ; pada input



Gambar 3.3: Alur kerja tokenisasi

### 3.4 Implementasi *Parser*

Proses parsing adalah proses mengorganisir seluruh token menjadi sebuah representasi tengah agar dapat dibaca oleh eksekutor. Setelah input dirubah menjadi token dan disimpan, token-token tersebut dipanggil satu persatu sesuai urutan lalu dibaca nilainya untuk kemudian diterjemahkan menjadi kode program yang merepresentasikan input perintah. Pada pembuatan interpreter INI, proses parsing dilakukan dengan menggunakan *switch-case* antara nilai token dengan syntax. Jika ditemukan syntax yang cocok untuk suatu nilai token, maka nilai token tersebut akan diterjemahkan ke dalam

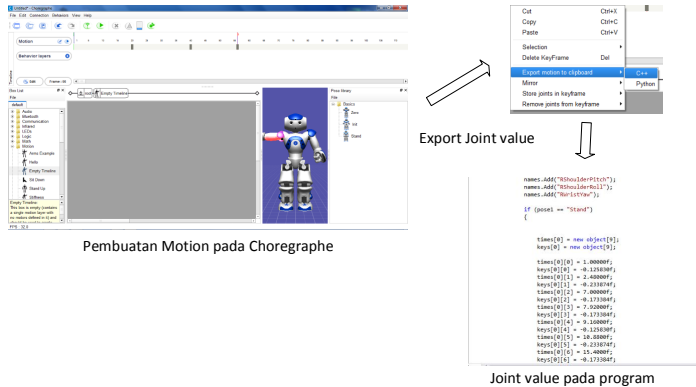
bentuk kode programnya sebagai representasi tengah dari masukan perintah untuk kemudian dieksekusi. Namun, jika pada suatu nilai token tidak ditemukan nilai syntax yang cocok, maka masukan dikatakan tidak sesuai syntax dan dikeluarkan keterangan "syntax error" pada *user interface*. Secara lebih detail, proses parsing dapat diamati Gambar 2.12

Dalam pembuatan interpreter INI, penulis menggunakan dua jenis cara yang berbeda untuk membuat kode program hasil parsing yang akan dieksekusi untuk menggerakkan robot NAO. Cara tersebut dibedakan sesuai dengan tipe perintah yang ada pada syntax. Tipe perintah tersebut adalah perintah dengan parameter dan perintah tanpa parameter. Pada perintah tanpa parameter seluruh data interpolasi joint sudah ditentukan sebelumnya di dalam kode program, sehingga saat user memanggil gerakan ini maka robot akan bergerak sesuai gerakan yang telah ditentukan. Sedangkan pada perintah dengan parameter, pengguna menentukan seberapa besar parameter yang dilakukan oleh sebuah gerakan. Kemudian nilai besaran tersebut di teruskan ke dalam kode program sehingga eksekusi perintah yang terjadi dipengaruhi oleh besar satuan yang dimasukkan oleh user. Selain parser untuk gerakan, di dalam interpreter INI juga terdapat parser untuk perulangan, parser untuk perintah parallel, dan parser untuk pengaturan kecepatan.

### 3.4.1 Perintah tanpa parameter

Gerakan tanpa parameter adalah gerakan yang dipanggil dengan perintah input yang tidak memiliki parameter. Pada saat gerakan jenis ini dipanggil, robot NAO akan melakukan gerakan tertentu yang sudah ditentukan. pada Tabel 3.2 dapat dilihat list dari gerakan tanpa parameter yang ada pada program interpreter INI.

Kode program untuk gerakan tanpa parameter dibuat dengan menggunakan data pergerakan interpolasi dari *joint* yang didapatkan dari Choregraphe. Tahapan pembuatannya dapat dilihat pada Gambar 3.4. Sebagian dari file gerakan choregraphe didapatkan dari sebuah website yang dikelola oleh *F.U.N lab* dari *University of Notre Dame*[14]. File *crg* berisi gerakan yang didapat dari website tersebut kemudian dimodifikasi seperlunya dan dimasukkan kedalam program INI. Perintah yang ditebalkan merupakan perintah yang dimodifikasi dari *crg* yang didapat dari website *F.U.N Lab*.



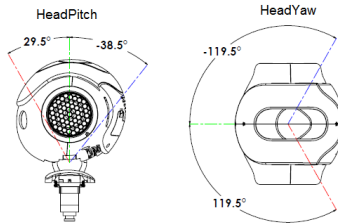
Gambar 3.4: Implementasi perintah tanpa parameter

Untuk gerakan yang dibuat sendiri, tahapan yang dilakukan ya-  
itu:

1. Dibuat sebuah *behavior* menggunakan Choregraphe, misal: *behavior* duduk, *behavior* berdiri, *behavior* hormat, dan sebagainya.
2. Kemudian, dari *behavior* yang dibuat pada Choregraphe tersebut data perubahan nilai *joint* dari waktu ke waktu di ekspor ke dalam bahasa pemrograman python, karena pada choregraphe hanya support ekspor untuk python dan c++.
3. Dilakukan sedikit modifikasi yaitu dengan penambahan tipe data, merubah struktur kode, dan perubahan simbol yang digunakan sehingga data interpolasi yang semula digunakan untuk bahasa pemrograman python bisa support untuk .Net. Setelah itu nilai *joint* tersebut dimasukkan ke dalam program untuk dapat dieksekusi jika di panggil oleh interpreter.

### 3.4.2 Perintah dengan parameter

Kode program untuk perintah dengan parameter dibuat dengan memasukkan variabel parameter yang didapat dari masukan perintah ke dalam kode program yang dijalankan. Sebagai contoh pada perintah **jalan maju [jarak]**, parameter jarak yang dimasukan oleh

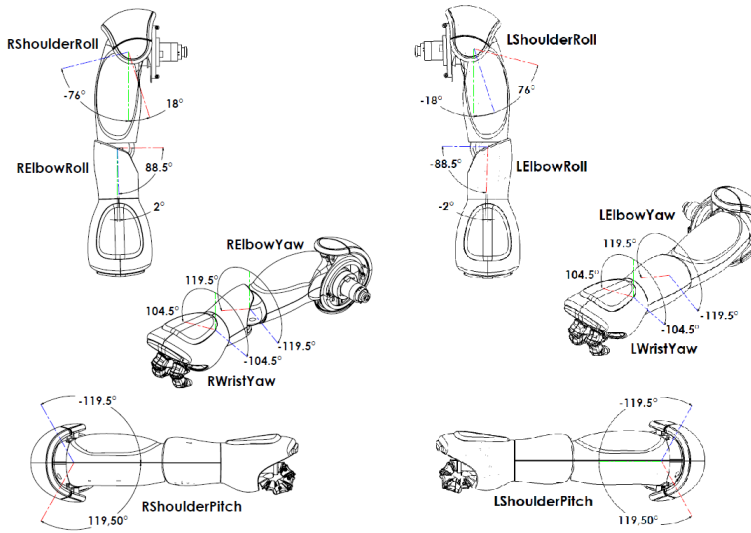


Gambar 3.5: Batas Sudut Kepala [3]

user disimpan dan kemudian dimasukkan ke dalam kode program yang mengeksekusi gerakan jalan sehingga robot NAO bergerak sesuai dengan jarak yang di inputkan oleh user. Daftar perintah dengan parameter dapat dilihat pada tabel 3.3. Namun demikian, perlu dilakukan sedikit modifikasi agar satuan dari parameter yang ada pada syntax lebih mudah digunakan oleh user. Modifikasi yang dilakukan antara lain:

1. Pada parameter [jarak], robot NAO memiliki modul *motion* untuk berjalan dengan perintah *WalkTargetVelocity* yang menerima parameter kecepatan sebagai masukannya. Dilakukan modifikasi perintah dengan memasukkan kecepatan maksimum untuk perintah tersebut. Dengan kecepatan maksimum, nao menempuh jarak 9.52cm dalam satu detiknya. Dengan perintah tersebut jika user memasukkan input jarak 1 meter, maka kode program akan mematikan perintah *WalkTargetVelocity* setelah 100/9.52 detik.
2. Pada parameter [arah] yang digunakan pada syntax tertentu, robot NAO menerima nilai sudut dalam satuan radian sehingga nilai parameter arah dalam parameter derajat oleh user dirubah menjadi radian.

Selain itu input parameter arah yang dimasukkan oleh user juga dinormalisasi jika input tersebut melebihi batas kemampuan aktuator robot NAO. Pada Tabel 3.4 dapat dilihat batas-batas yang dimaksud. Pembatasan tersebut dilakukan untuk menyesuaikan dengan spesifikasi robot NAO. Pada Gambar 3.5 dan 3.6 dapat dilihat batas sudut aktuator pada robot NAO.



Gambar 3.6: Batas Sudut Tangan [3]

### 3.4.3 Perulangan

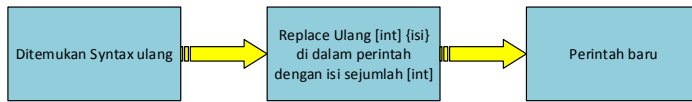
Syntax perulangan pada interpreter INI memiliki aturan sebagai berikut:

ulang [integer] {[perintah yang diulang]}

*Parser* pada syntax perulangan dibuat dengan menggunakan *regex* untuk mengedit masukan. Jika ditemukan syntax perulangan, maka masukan akan diubah dengan menggunakan *regex* untuk menuliskan perintah yang terdapat di dalam tanda kurung kurawal sebanyak jumlah perulangan yang diinginkan. Sebagai contoh: User memasukkan perintah ulang 2 {duduk; berdiri;}. Sehingga di dalam hasil *parser* akan dituliskan seolah-olah pengguna menginputkan perintah duduk; berdiri; duduk; berdiri;. Proses yang dilakukan untuk memproses syntax perulangan dapat diamati pada Gambar 3.7.

### 3.4.4 *Parallel processing*

Interpreter INI dapat menjalankan dua perintah secara bersamaan agar didapatkan dua gerakan yang terjadi bersamaan. Syntax



Gambar 3.7: Proses pada syntax perulangan

*parallel processing* pada interpreter INI memiliki aturan sebagai berikut: ([perintah yang diparalel])

Parser pada syntax paralel processing dibuat dengan menggunakan *thread* baru untuk setiap perintah yang diproses. Jika ditemukan syntax paralel maka setiap perintah didalam tanda kurung akan di eksekusi dengan thread tersendiri. Sebagai contoh: User ingin-putkan perintah (angkat lengan kanan keatas 90; jalan maju 1;). Pada saat dieksekusi, angkat lengan kanan keatas 90; dan jalan maju 1; akan dieksekusi oleh *thread* yang berbeda sehingga gerakan tersebut terjadi hampir bersamaan. Pada kasus dimana *joint* yang digerakkan antara dua *thread* memiliki kesamaan, maka akan dikeluarkan notifikasi "syntax error".

### 3.4.5 Pengaturan kecepatan

Syntax kecepatan pada interpreter INI memiliki aturan sebagai berikut: kecepatan [integer antara 1 sampai 5]

Semakin besar nilai integer yang dibuat, maka akan semakin cepat suatu eksekusi gerakan. Kecepatan awal pada interpreter INI adalah 4. Parser pada syntax kecepatan dibuat dengan merubah-ubah variabel waktu yang ada pada eksekutor. Variabel tersebut mempengaruhi lama waktu yang diperlukan NAO untuk melakukan sebuah gerakan. Pengaturan kecepatan pada interpreter INI untuk perintah dengan parameter ditetapkan sebagai berikut:

1. Parameter 1, kecepatan eksekusi robot NAO 5 detik.
2. Parameter 2, kecepatan eksekusi robot NAO 4 detik.
3. Parameter 3, kecepatan eksekusi robot NAO 3 detik.
4. Parameter 4, kecepatan eksekusi robot NAO 2 detik.
5. Parameter 5, kecepatan eksekusi robot NAO 1 detik.



Sebagai contoh user memasukkan perintah kecepatan 5; angkat lengan kanan keatas 90; kecepatan 1; angkat lengan kanan kebawah 90;. Maka robot NAO akan bergerak mengangkat lengan kanannya keatas dalam waktu 1 detik dan menurunkannya dalam waktu 5 detik.

Untuk perintah tanpa parameter, parameter kecepatan diimplementasikan dengan merubah-rubah array waktu yang ada pada gerakan tersebut. Sebuah perintah tanpa parameter memiliki array waktu standar, yang ditetapkan sebagai array waktu dimana gerakan tersebut dibuat. Setiap nilai yang ada pada array waktu ini kemudian ditambahkan dengan variabel berikut:  $(4 - \text{kecepatan}) * ((\text{nilai array standar sekarang} - \text{nilai array standar sebelumnya}) / 4)$ . Sebagai contoh user memasukkan perintah kecepatan 5; duduk;. Maka setiap array waktu yang ada dalam perintah duduk akan dilakukan penjumlahan dengan variabel  $-1 * ((\text{nilai array standar sekarang} - \text{nilai array standar sebelumnya}) / 4)$

### 3.4.6 Pengucapan kalimat

Pengucapan kalimat pada interpreter INI dapat dilakukan dengan syntax katakana. Syntax katakana pada interpreter INI memiliki aturan sebagai berikut: katakana "[isi dari pesan yang ingin dikatakan oleh NAO]"

Isi dari pesan adalah sebuah string berbahasa inggris yang nantinya akan di ucapkan oleh robot NAO. Robot NAO yang dimiliki lab saat ini hanya dapat menggunakan bahasa inggris sebagai bahasa utama, sehingga disini penulis memasukkan syntax katakana untuk membuat robot NAO mengucapkan pesan dalam bahasa inggris. Parser pada syntax ini dibuat dengan mengisikan variabel string yang dimasukkan oleh user ke dalam kode program yang memanggil modul Naoqi yang mengontrol bagian bicara yaitu modul *AlTextToSpeech*.

## 3.5 Eksekutor

Proses selanjutnya adalah proses eksekusi dari hasil proses parsing. Proses ini dijalankan oleh bagian dari interpreter yang disebut eksekutor. Di bagian eksekutor ini kode program untuk menggerakkan robot NAO di eksekusi. Setelah kode program dieksekusi, program akan mengirimkan perintah kepada Naoqi yang ada di robot

untuk menggerakkan aktuator yang ada di robot. Proses gerak dari aktuator setelah perintah dikirimkan ke robot NAO sepenuhnya di kontrol oleh Naoqi yang ada di dalam robot NAO. Dalam program interpreter INI, tidak seluruh modul yang ada di Naoqi digunakan di dalam interpreter INI. Hal ini dikarenakan modul di dalam Naoqi berjumlah banyak sehingga waktu untuk pengerjaan dirasa tidaklah cukup. Selain itu target utama penulis dalam mendesain Interpreter INI adalah pengontrolan gerakan, sehingga modul-modul yang berhubungan dengan sensor belum di masukan oleh penulis ke dalam Interpreter INI. Modul yang digunakan oleh penulis adalah: modul *AlMotion*, modul *AlRobotPose* dan modul *AlTextToSpeech*.

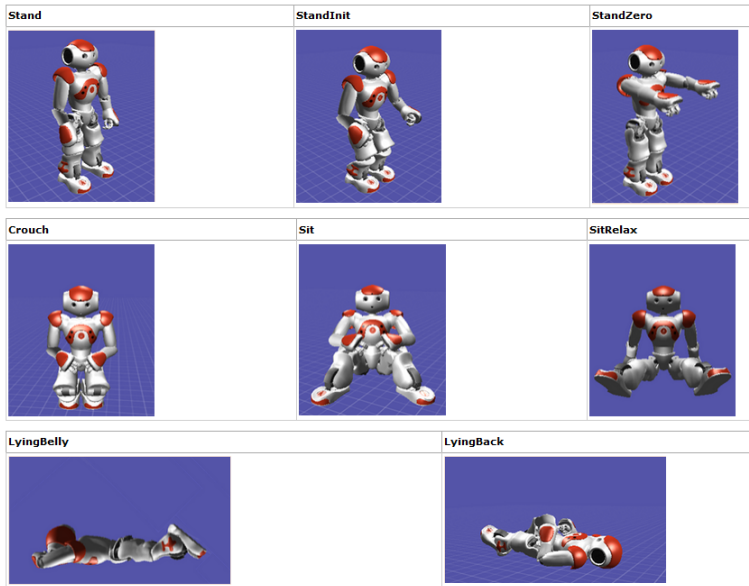
### 3.5.1 Eksekutor untuk perintah gerakan

Perintah yang berhubungan dengan gerakan dieksekusi dengan memberikan perintah kepada modul *AlMotion* pada Naoqi. Modul *AlMotion* adalah sebuah modul dari Naoqi yang memiliki fungsi mengontrol gerakan dari robot NAO. Didalamnya terdapat fungsi - fungsi yang dapat dipanggil untuk menjalankan gerakan pada robot NAO. Pada program interpreter INI, fungsi yang digunakan antara lain:

1. *AngleInterpolation*: Menginterpolasi satu atau lebih joint sesuai dengan target sudut tertentu dan trayek waktu tertentu yang ditentukan.
2. *AngleInterpolationBezier*: Menginterpolasi sequence sudut yang memiliki waktu untuk beberapa motor menggunakan kontrol poin Bezier.
3. *SetStiffness*: Menetapkan Stiffness ke satu atau lebih joint
4. *setWalkTargetVelocity*: Membuat robot NAO bergerak dengan kecepatan yang ditentukan.
5. *walkTo*: Membuat robot NAO bergerak ke posisi yang ditentukan.

### 3.5.2 Eksekutor untuk deteksi pose pada robot NAO

Dalam menjalankan fungsinya, interpreter INI perlu untuk mendeteksi pose yang sedang dilakukan oleh robot NAO. Untuk itu interpreter INI menggunakan modul *AlRobotPose* yang ada pada robot NAO. Modul *AlRobotPose* adalah modul dari Naoqi yang me-



Gambar 3.8: Pose robot NAO[3]

memiliki fungsi mengestimasi pose yang sedang dilakukan oleh robot NAO. Sebuah pose dapat dikatakan sebagai klasifikasi dari kondisi *joint-joint* secara keseluruhan dari robot NAO. Pose-pose yang ada di dalam robot NAO dapat dilihat pada Gambar 3.8. Pada interpreter INI pose digunakan untuk mengetahui posisi dari robot sebelum mengeksekusi suatu gerakan tertentu yang memerlukan posisi awal. Sebagai contoh saat dipanggil perintah berdiri, maka *AlRobotPose* dapat diketahui pose robot saat itu. Gerakan berdiri dari posisi duduk dan gerakan berdiri dari pose jongkok tentunya menggunakan *interpolasi joint* yang berbeda.

### 3.5.3 Eksekutor Untuk Pengucapan

Pada syntax pengucapan, interpreter ini menggunakan modul *AlTextToSpeech* pada robot NAO. Modul *AlTextToSpeech* adalah modul dari Naoqi yang memiliki fungsi merubah text yang dimasukkan menjadi ucapan berbasis *Pulse Code Modulation (PCM)*

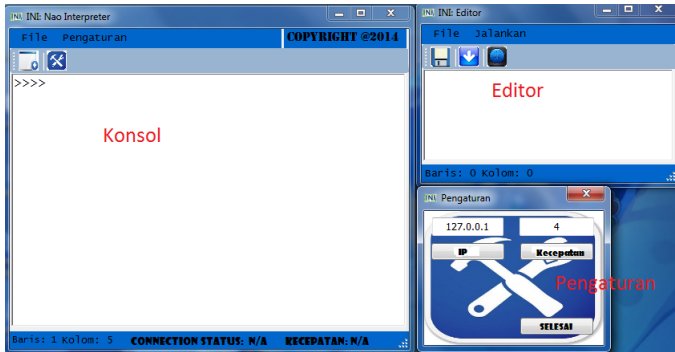
yang akan diputar di loudspeaker yang ada di robot NAO. Pada *AlTextToSpeech* sendiri terdapat beberapa pilihan bahasa yang disediakan oleh Aldebaran. Bahasa yang disediakan antara lain: *Chinese, English, French, German, Italian, Japanese, Korean, Portuguese*, dan *Spanish*. Tetapi bahasa yang dimiliki oleh robot NAO yang digunakan oleh penulis hanyalah bahasa inggris sehingga penulis hanya memasukkan modul *AlTextToSpeech* berbahasa inggris di dalam program interpreter INI.

### 3.6 User Interface

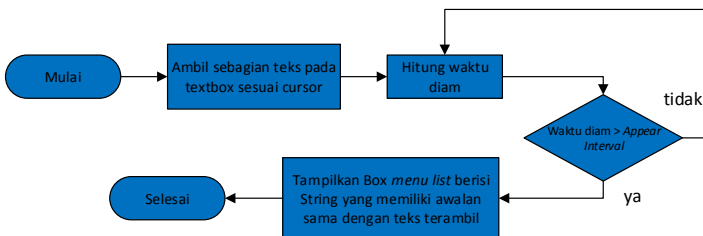
Proses selanjutnya adalah pembuatan user interface dari program interpreter INI. User interface yang dibuat memiliki beberapa fitur untuk mendukung program interpreter INI. Fitur-fitur tersebut antara lain:

1. *New Window*: Untuk membuka jendela kosong baru, yaitu jendela dimana user memasukkan input perintah yang akan di interpretasi.
2. *Save*: Untuk menyimpan teks yang ada pada jendela input. Penamaan nama jendela secara otomatis mengikuti nama file tempat teks disimpan.
3. *Load*: Untuk mengambil isi teks dari suatu file kemudian memasukkannya ke dalam jendela yang sedang di buka. Penamaan nama jendela secara otomatis mengikuti nama file tempat teks disimpan.
4. *Run*: Untuk memulai proses interpretasi input perintah yang telah di masukkan pada jendela input.
5. *Pengaturan*: untuk membuka window pengaturan dimana didalamnya user dapat mengatur ip dan juga kecepatan gerakan dari robot NAO.

*User interface* secara lengkap dapat diamati pada Gambar 3.9. Bagian *Console* merupakan bagian utama dari interface yang menampilkan notifikasi dan juga dapat digunakan untuk memasukkan perintah. Bagian *editor* adalah bagian dari interpreter yang digunakan untuk memasukkan baris-baris perintah yang akan digunakan untuk memerintahkan robot NAO. Pada bagian *editor* pengguna juga dapat menyimpan baris-baris perintah yang telah dibuat sehingga dapat dibuka kembali di lain waktu. Pada bagian pengaturan



Gambar 3.9: *User Interface*



Gambar 3.10: Diagram alur *Auto Completer*

pengguna dapat mengatur alamat ip robot yang ingin diperintah menggunakan interpreter INI. Pada panel *editor* dan panel *console* juga dilengkapi dengan *Auto Completer* buatan Pavel Torgashov untuk mengurangi kesalahan penulisan syntax. Diagram alur dari *Auto Completer* tersebut secara umum dapat dilihat pada Gambar 3.10.

Tabel 3.2: Perintah tanpa parameter

<b>Nama perintah</b>	<b>Keterangan</b>
berdiri	robot NAO berdiri dari posisi awal duduk, jongkok
<b>cium</b>	robot NAO melakukan gerakan cium jauh
<b>dansa tangan</b>	robot NAO melakukan gerakan dansa dengan tangan
<b>dansa thriller</b>	robot NAO melakukan gerakan dansa thriller
<b>dansa disco</b>	robot NAO melakukan gerakan dansa disco
<b>dansa pesta</b>	robot NAO melakukan gerakan dansa saat pesta
<b>dansa macarena</b>	robot NAO melakukan gerakan dansa macarena
duduk	robot NAO duduk dari posisi awal berdiri
hormat	robot NAO melakukan gerakan hormat
jongkok	robot NAO jongkok dari posisi awal berdiri
<b>kecapekan</b>	robot NAO mengusap dahi
<b>membungkuk</b>	robot NAO melakukan gerakan membungkuk
<b>menantang</b>	robot NAO menantang lawan
<b>main gitar</b>	robot NAO menggitar air gitar
<b>pushup</b>	robot NAO melakukan gerakan pushup
<b>senam</b>	robot NAO melakukan gerakan senam
siap	robot NAO menuju posisi siap untuk tangan
taichi	robot NAO melakukan gerakan senam taichi
salam	robot NAO melambaikan tangan
<b>menunjuk</b>	robot NAO menunjuk jari ke depan
setuju	robot NAO menganggukan kepala tanda setuju
geleng	robot NAO menggelengkan kepala tidak setuju
tepuk tangan	robot NAO bertepuk tangan
pukul	robot NAO memukul lawan
lindungi kepala	robot NAO melindungi kepala
tendang	robot NAO menendang
tangkis	robot NAO melakukan tepisan
seruduk	robot NAO menyeruduk
goyang	robot NAO menggoyangkan pinggul
pemanasan	robot NAO melakukan pemanasan
berdoa	robot NAO berdoa
menangis	robot NAO berekspresi menangis
tertawa	robot NAO berekspresi tertawa
marah	robot NAO berekspresi marah
pose berfoto	robot NAO melakukan pose berfoto

Tabel 3.3: Perintah dengan parameter

<b>Nama perintah</b>	<b>Parameter</b>
tengok	[kanan/kiri/atas/bawah/depan]
hadap	[kanan/kiri]
serong	[kanan/kiri]
balik	[kanan/kiri]
jalan	[maju/mundur/kanan/kiri] [jarak]
langkah	[kanan/kiri] [arah]
maju	[jumlah langkah]
mundur	[jumlah langkah]
angkat kaki	[kanan/kiri]
tangan	[kanan/kiri/kanan dan kiri] [keatas/ke depan]
lencang	[kanan/kiri/depan]
condong	[arah]
jari	[dibuka/ditutup]

Tabel 3.4: Batas sudut

<b>Nama perintah</b>	<b><i>Range (degrees)</i></b>
tengok kanan/kiri	119.5/119.5
tengok atas/bawah	38.5/29.5
gerakan lengan keatas/kebawah	119.5/119.5
gerakan lengan kanan kekanan/kekiri	76/18
gerakan lengan kiri kekanan/kekiri	18/76
putar hasta kanan/kiri	-104.5/104.5

## **BAB 4**

### **PENGUJIAN DAN ANALISA**

Pada bab ini akan dipaparkan hasil pengujian dan analisa dari desain sistem dan implementasi yang sudah dibahas pada bab 3. Bentuk pengujian yang dilakukan penulis untuk menguji interpreter INI adalah sebagai berikut.

#### **4.1 Pengujian syntax**

Pada pengujian masukan perintah, hal yang diamati adalah bagaimana interpreter INI merespon ketika perintah dimasukkan dengan berbagai kondisi masukan yang berbeda. Pada Bab 4.1.1 akan dilakukan pengujian interpreter INI dengan menggunakan syntax input yang benar. Pada Bab 4.2 akan dilakukan pengujian interpreter INI dengan menggunakan syntax input yang salah

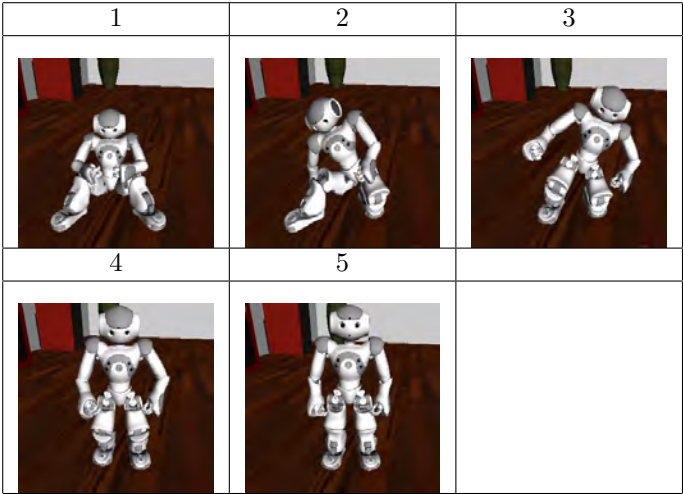
##### **4.1.1 Syntax benar**

Pada pengujian ini, Interpreter diberi masukan berupa input perintah dengan syntax yang benar. Syntax yang benar bagi perintah tanpa parameter adalah menuliskan tiap perintah dengan diakhiri tanda (;) untuk setiap perintahnya. Misal, duduk; berdiri; jongkok; hormat; berdiri; dansa1; dan sebagainya. Berikut ini adalah hasil dari pengujian dari perintah tanpa parameter dengan syntax input benar.

1. Tabel 4.1 memperlihatkan eksekusi robot NAO ketika syntax perintah berdiri yang dimasukkan ke interpreter benar.
2. Tabel 4.2 memperlihatkan eksekusi robot NAO ketika syntax perintah yang dimasukkan ke interpreter benar.
3. Tabel 4.3 memperlihatkan eksekusi robot NAO ketika syntax perintah cium yang dimasukkan ke interpreter benar.
4. Tabel 4.4 memperlihatkan eksekusi robot NAO ketika syntax perintah dansa1 yang dimasukkan ke interpreter benar.
5. Tabel 4.5 memperlihatkan eksekusi robot NAO ketika syntax perintah duduk yang dimasukkan ke interpreter benar.
6. Tabel 4.6 memperlihatkan eksekusi robot NAO ketika syntax perintah hormat yang dimasukkan ke interpreter benar.
7. Tabel 4.7 memperlihatkan eksekusi robot NAO ketika syntax perintah jongkok yang dimasukkan ke interpreter benar.



Tabel 4.1: Gerakan berdiri dengan syntax masukan benar



8. Tabel 4.8 memperlihatkan eksekusi robot NAO ketika syntax perintah kecapekan yang dimasukkan ke interpreter benar.

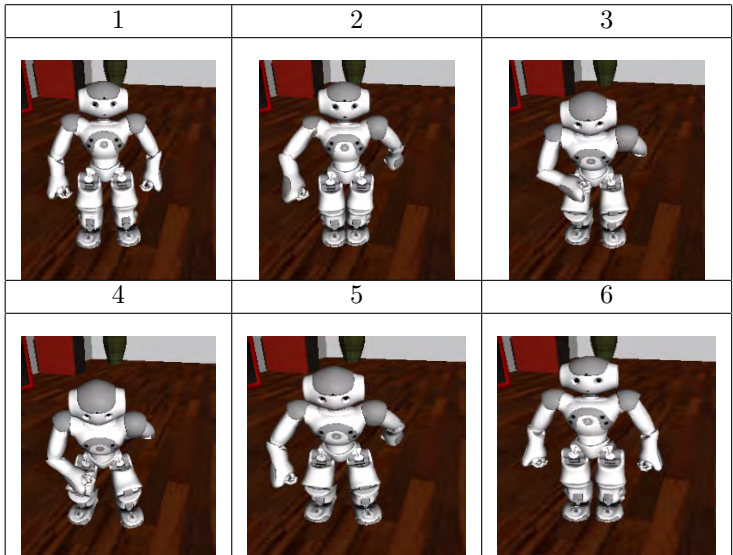
Pada perintah dengan parameter, syntax yang benar adalah dengan melengkapi perintah sesuai parameter yang dibutuhkan dan mengakhirinya dengan tanda (;). Misal, jalan maju 1; jalan mundur 1; angkat lengan kanan keatas 90; dan lain sebagainya. Berikut ini adalah hasil pengujian dari perintah dengan parameter dengan syntax input benar.

## 4.2 Syntax salah

Pada pengujian ini, interpreter diberi masukan berupa input perintah dengan syntax yang salah. Pengujian dengan syntax yang salah diperlukan karena ada kemungkinan user memasukkan input yang tidak sesuai syntax ke dalam masukan dari interpreter. Berikut adalah hasil pengujian interpreter ketika diberi masukan input dengan syntax yang salah berdasarkan tipe kesalahan yang dilakukan:

1. Kesalahan berupa penulisan syntax, yaitu jika user memasukkan perintah yang tidak dikenali. Sebagai contoh: beradi,

Tabel 4.2: Gerakan membungkuk dengan syntax masukan benar



cuim, das, jaln maju 1, sdss, dansa8, dasss, zzzzz, dan hrmeat. Kesalahan berupa penulisan syntax dapat dilihat pada tabel 4.11

2. Kesalahan berupa penulisan parameter, yaitu pada saat memasukkan perintah dengan parameter tipe data dari parameter yang dimasukkan tidak sesuai syntax. Kesalahan berupa penulisan parameter dapat dilihat pada tabel 4.12
3. Kesalahan berupa tidak menambahkan tanda (;) pada akhir perintah dapat dilihat pada tabel 4.13

### 4.3 Pengujian keseimbangan robot

Pada pergerakan robot NAO, dilakukan pengujian mengenai faktor keseimbangan yang dapat hilang sewaktu robot melakukan gerakan. Hilangnya keseimbangan dari robot dapat terjadi karena beberapa faktor, antara lain:

1. Pergerakan robot yang terlalu cepat sehingga menghasilkan momentum yang dapat membuat robot kehilangan keseim-

- bagian
2. Kombinasi dari 2 buah gerakan yang menyebabkan robot kehilangan keseimbangan

#### **4.3.1 Pengujian kecepatan terhadap keseimbangan**

Pengujian pengaruh kecepatan terhadap keseimbangan dilakukan dengan menguji beberapa gerakan robot yang dianggap dapat mempengaruhi keseimbangan dengan kecepatan yang berbeda-beda. Jika robot terjatuh maka dapat dikatakan gerakan yang dilakukan robot terlalu cepat sehingga perlu dilakukan pengurangan kecepatan dalam jumlah tertentu hingga didapatkan kondisi stabil dimana kecepatan dalam pelaksanaan gerakan tersebut tidak menyebabkan robot terjatuh. Berikut ini adalah hasil dari pengujian dari perintah tanpa parameter dengan berbagai level kecepatan.

1. Tabel 4.14-4.17 memperlihatkan eksekusi gerakan duduk dengan kecepatan 4 sampai dengan kecepatan 7. Dari hasil pengujian didapatkan bahwa NAO jatuh pada saat melakukan gerakan duduk dengan nilai kecepatan 7.
2. Tabel 4.18-4.21 memperlihatkan eksekusi gerakan berdiri dengan kecepatan 4 sampai dengan kecepatan 7. Dari hasil pengujian didapatkan bahwa NAO jatuh pada saat melakukan gerakan berdiri dengan nilai kecepatan 7.
3. Tabel 4.22-4.25 memperlihatkan eksekusi gerakan berdiri dengan kecepatan 4 sampai dengan kecepatan 7. Dari hasil pengujian didapatkan bahwa NAO jatuh pada saat melakukan gerakan taichi dengan nilai kecepatan 7.









#### **4.3.2 Pengujian kombinasi perintah terhadap keseimbangan**

Pengujian pengaruh kombinasi gerakan terhadap keseimbangan dilakukan dengan menguji robot dengan melakukan kombinasi gerakan robot yang dianggap mempengaruhi keseimbangan dari robot. Pengujian dimulai dengan memberikan perintah gerakan dengan nilai paling minimum yang mempengaruhi keseimbangan robot. Kemudian nilai tersebut ditambah secara bertahap hingga didapatkan nilai maksimum dari kombinasi perintah tersebut yang membuat robot terjatuh. Pada pengujian ini, robot diatur pada posisi awal










pose init. Berikut ini adalah hasil dari pengujian beberapa macam kombinasi gerakan:

1. Tabel 4.27, memperlihatkan kombinasi condong kebelakang dan angkat lengan kanan keatas dengan parameter sudut yang berbeda-beda. Dari hasil pengujian didapatkan bahwa pada saat parameter angkat lengan 105 derajat robot NAO terjatuh.
2. Tabel 4.26, memperlihatkan kombinasi condong kebelakang dan angkat lengan kiri keatas dengan parameter sudut yang berbeda-beda. Dari hasil pengujian didapatkan bahwa pada saat parameter angkat lengan 105 derajat robot NAO terjatuh.
3. Tabel 4.28, memperlihatkan kombinasi angkat kaki kanan dan angkat lengan kiri keatas dengan parameter sudut yang berbeda-beda. Dari hasil pengujian didapatkan bahwa pada meskipun sudut angkat lengan kiri sudah mencapai 150 derajat namun NAO tetap tidak jatuh.
4. Tabel 4.29, memperlihatkan kombinasi angkat kaki kiri dan angkat lengan kanan keatas dengan parameter sudut yang berbeda-beda. Dari hasil pengujian didapatkan bahwa pada meskipun sudut angkat lengan kiri sudah mencapai 150 derajat namun NAO tetap tidak jatuh.









Tabel 4.3: Gerakan cium dengan syntax masukan benar

1	2	3
		
4	5	6
		
7	8	
		

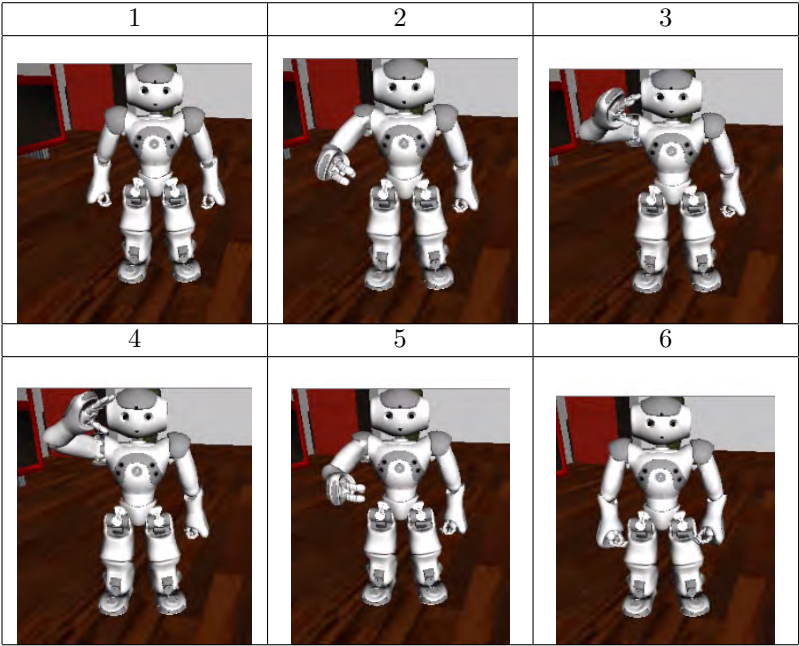
Tabel 4.4: Gerakan dansa tangan dengan syntax masukan benar

1	2	3
		
4	5	6
		
7	8	9
		

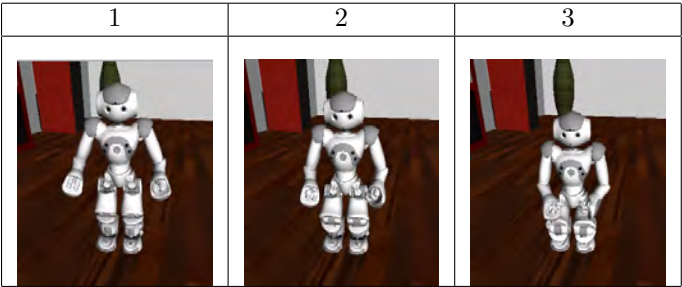
Tabel 4.5: Gerakan duduk dengan syntax masukan benar

1	2	3
		
4	5	6
		
7	8	
		

Tabel 4.6: Gerakan hormat dengan syntax masukan benar











Tabel 4.7: Gerakan jongkok dengan syntax masukan benar











Tabel 4.8: Gerakan kecapekan dengan syntax masukan benar

1	2	3
		
4	5	6
		
7	8	
		

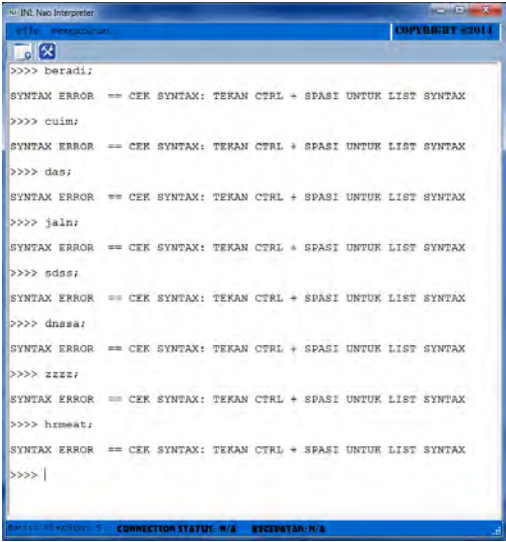
Tabel 4.9: Gerakan hadap kanan dengan syntax benar

1	2	3
		

Tabel 4.10: Gerakan tengok kanan syntax masukan benar

1	2	3
		


Tabel 4.11: Kesalahan berupa penulisan syntax

Contoh kesalahan	Keluaran pada console
<pre>beradi; cuim; das; jaln; sdss; dnssa; dasss; zzzzz; hrmeat;</pre>	 <pre>JPL Neo Interpreter File Edit Window Help http://www.jpl.com COPYRIGHT 2014  &gt;&gt;&gt;&gt; beradi; SYNTAX ERROR == CEK SYNTAX: TEKAN CTRL + SPASI UNTUK LIST SYNTAX &gt;&gt;&gt;&gt; cuim; SYNTAX ERROR == CEK SYNTAX: TEKAN CTRL + SPASI UNTUK LIST SYNTAX &gt;&gt;&gt;&gt; das; SYNTAX ERROR == CEK SYNTAX: TEKAN CTRL + SPASI UNTUK LIST SYNTAX &gt;&gt;&gt;&gt; jaln; SYNTAX ERROR == CEK SYNTAX: TEKAN CTRL + SPASI UNTUK LIST SYNTAX &gt;&gt;&gt;&gt; sdss; SYNTAX ERROR == CEK SYNTAX: TEKAN CTRL + SPASI UNTUK LIST SYNTAX &gt;&gt;&gt;&gt; dnssa; SYNTAX ERROR == CEK SYNTAX: TEKAN CTRL + SPASI UNTUK LIST SYNTAX &gt;&gt;&gt;&gt; zzzzz; SYNTAX ERROR == CEK SYNTAX: TEKAN CTRL + SPASI UNTUK LIST SYNTAX &gt;&gt;&gt;&gt; hrmeat; SYNTAX ERROR == CEK SYNTAX: TEKAN CTRL + SPASI UNTUK LIST SYNTAX &gt;&gt;&gt;&gt;    Help: http://jpl.com 5 CONNECTION STATUS: N/A REGISTRATION: N/A</pre>





Tabel 4.12: Kesalahan berupa penulisan parameter

Contoh kesalahan	Keluaran pada console
<p>                     hadap dfs;                      serong jlj;                      jalan lklk;                      maju kjkj;                      lencang kjkj;                      tangan kjkj;                          dansa;                      dansa kkk;                 </p>	 <pre> P0 NaoInterpreter File: Program.cs COPYRIGHT 2014  &gt;&gt;&gt;&gt; hadap dfs; SYNTAX ERROR == CEK SYNTAX: hadap [kanan/kiri] &gt;&gt;&gt;&gt; serong jlj; SYNTAX ERROR == CEK SYNTAX: serong [kanan/kiri] &gt;&gt;&gt;&gt; jalan lklk; SYNTAX ERROR == CEK SYNTAX: jalan [arah] [jarak(meter)] &gt;&gt;&gt;&gt; maju kjkj; SYNTAX ERROR == CEK SYNTAX: maju [jumlah langkah] &gt;&gt;&gt;&gt; lencang kjkj; SYNTAX ERROR == CEK SYNTAX: lencang [kanan/kiri/depan] &gt;&gt;&gt;&gt; tangan kjkj; SYNTAX ERROR == CEK SYNTAX: tangan [kanan/kiri/kanan dan kiri] [kedo] &gt;&gt;&gt;&gt; dansa; SYNTAX ERROR == CEK SYNTAX: dansa [tangan/thriller/disco/pesta/macare] &gt;&gt;&gt;&gt; dansa kkk; SYNTAX ERROR == CEK SYNTAX: dansa [tangan/thriller/disco/pesta/macare] &gt;&gt;&gt;&gt;     </pre>




Tabel 4.13: Kesalahan berupa tidak menambahkan tanda (;) pada akhir perintah

Contoh kesalahan	Keluaran pada console
main gitar senam duduk berdiri dansa tangan dansa macarena kecapekan cium siap	 <pre> &gt;&gt;&gt; main gitar SYNTAX ERROR == JANGAN LUPA MENAMBAHKAN TITIK KOMA ( ; ) PADA AKHIR PERINTAH &gt;&gt;&gt; senam SYNTAX ERROR == JANGAN LUPA MENAMBAHKAN TITIK KOMA ( ; ) PADA AKHIR PERINTAH &gt;&gt;&gt; duduk SYNTAX ERROR == JANGAN LUPA MENAMBAHKAN TITIK KOMA ( ; ) PADA AKHIR PERINTAH &gt;&gt;&gt; berdiri SYNTAX ERROR == JANGAN LUPA MENAMBAHKAN TITIK KOMA ( ; ) PADA AKHIR PERINTAH &gt;&gt;&gt; dansa tangan SYNTAX ERROR == JANGAN LUPA MENAMBAHKAN TITIK KOMA ( ; ) PADA AKHIR PERINTAH &gt;&gt;&gt; dansa macarena SYNTAX ERROR == JANGAN LUPA MENAMBAHKAN TITIK KOMA ( ; ) PADA AKHIR PERINTAH &gt;&gt;&gt; kecapekan SYNTAX ERROR == JANGAN LUPA MENAMBAHKAN TITIK KOMA ( ; ) PADA AKHIR PERINTAH &gt;&gt;&gt; cium SYNTAX ERROR == JANGAN LUPA MENAMBAHKAN TITIK KOMA ( ; ) PADA AKHIR PERINTAH &gt;&gt;&gt; siap SYNTAX ERROR == JANGAN LUPA MENAMBAHKAN TITIK KOMA ( ; ) PADA AKHIR PERINTAH </pre>




Tabel 4.14: Gerakan duduk dengan kecepatan standar

2 detik	4 detik	6 detik
		
8 detik		
		





Tabel 4.15: Gerakan duduk dengan kecepatan 5

2 detik	4 detik	6 detik
		





Tabel 4.16: Gerakan duduk dengan kecepatan 6

1 detik	2 detik	4 detik
		




Tabel 4.17: Gerakan duduk dengan kecepatan 7

0,5 detik	1 detik	1,5 detik
		
2 detik		
		

Tabel 4.18: Gerakan berdiri dengan kecepatan standar

2 detik	4 detik	6 detik
		
8 detik		
		

Tabel 4.19: Gerakan berdiri dengan kecepatan 5




2 detik	4 detik	6 detik
		

4.4 Survei Penggunaan Interpreter





Pada program Interpreter INI dilakukan survei pada 10 orang anak SD, dan SMP untuk mencoba menggunakan interpreter INI untuk membuat gerakan pada robot NAO. Dari hasil survei, sepuluh anak yang mencoba Interpreter INI mampu membuat gerakan pada robot NAO.












Tabel 4.20: Gerakan berdiri dengan kecepatan 6

1 detik	2 detik	4 detik
		










Tabel 4.21: Gerakan berdiri dengan kecepatan 7

0,5 detik	1 detik	1,5 detik
		
2 detik		
		










Tabel 4.22: Gerakan taichi dengan kecepatan standar

5 detik	10 detik	15 detik
		
20 detik	25 detik	30 detik
		
35 detik	40 detik	45 detik
		





Tabel 4.23: Gerakan taichi dengan kecepatan 5

4 detik	7.75 detik	11.5 detik
		
15.4 detik	19 detik	22.7 detik
		
26.6 detik	29.8 detik	34.1 detik
		







Tabel 4.24: Gerakan taichi dengan kecepatan 6

3 detik	5.1 detik	7.9 detik
		
10.5 detik	12.7 detik	15.3 detik
		
18 detik	20.1 detik	22.8 detik
		







Tabel 4.25: Gerakan taichi dengan kecepatan 7

1.5 detik	2.7 detik	3.4 detik
		
4 detik		
		










Tabel 4.26: Condong ke belakang + angkat lengan kiri

30 derajat	45 derajat	60 derajat
		
75 derajat	90 derajat	105 derajat
		

Tabel 4.27: Condong ke belakang + angkat lengan kanan





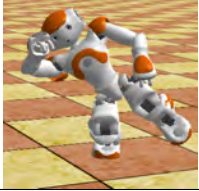




30 derajat	45 derajat	60 derajat
		
75 derajat	90 derajat	105 derajat
		

Tabel 4.28: angkat kaki kanan + angkat lengan kiri

30 derajat	45 derajat	60 derajat
		
75 derajat	90 derajat	105 derajat
		
120 derajat	135 derajat	150 derajat
		



Tabel 4.29: angkat kaki kiri + angkat lengan kanan

30 derajat	45 derajat	60 derajat
		
75 derajat	90 derajat	105 derajat
		
120 derajat	135 derajat	150 derajat
		



*Halaman ini sengaja dikosongkan*

# BAB 5

## PENUTUP

### 5.1 Kesimpulan

Dari hasil implementasi dan pengujian syntax yang dilakukan pada program interpreter INI yang sudah dilakukan dapat ditarik beberapa kesimpulan:

1. Interpreter yang telah dibuat jika diberikan masukan yang sesuai dengan syntax mampu menggerakkan robot NAO sesuai dengan pengertian dari perintah masukan. Ketika interpreter diberikan masukan yang tidak sesuai dengan syntax, interpreter sudah dapat memberikan notifikasi kesalahan. Namun notifikasi tersebut belum mampu memberi informasi lokasi dimana kesalahan tersebut terjadi.
2. Penggunaan interpreter memudahkan pengaturan gerakan robot NAO bagi anak kecil usia SD dan SMP karena tidak perlu menentukan jalur interpolasi yang akan dilakukan oleh setiap motor, melainkan hanya perlu memasukan sebuah perintah untuk pengaturan gerakan robot NAO.
3. Robot NAO kehilangan keseimbangan ketika melakukan gerakan taichi, gerakan berdiri, dan gerakan duduk dengan parameter kecepatan tujuh.
4. Robot NAO kehilangan keseimbangan pada kombinasi gerakan condong kebelakang + angkat lengan kanan/kiri keatas 105 derajat.
5. Dari 10 anak yang mencoba program Interpreter INI, semuanya mampu membuat gerakan untuk robot NAO.

### 5.2 Saran

Untuk pengembangan lebih lanjut mengenai tugas akhir ini, disarankan untuk melakukan beberapa langkah lanjutan:

1. Penambahan daftar gerakan baik untuk gerakan tanpa parameter dan gerakan dengan parameter.
2. Diperlukan mekanisme untuk mendeteksi lokasi dimana kesalahan terjadi pada masukan.
3. Penambahan modul Naoqi yang digunakan oleh eksekutor pada interpreter terutama modul sensor yang memiliki potensi

untuk membuat program menjadi lebih baik.

4. Keluaran konsol ketika terjadi kesalahan masukan dapat dibuat interaktif berupa pertanyaan kepada pengguna sehingga lebih menarik.

# BAB 5

## PENUTUP

### 5.1 Kesimpulan

Dari hasil implementasi dan pengujian syntax yang dilakukan pada program interpreter INI yang sudah dilakukan dapat ditarik beberapa kesimpulan:

1. Interpreter yang telah dibuat jika diberikan masukan yang sesuai dengan syntax mampu menggerakkan robot NAO sesuai dengan pengertian dari perintah masukan. Ketika interpreter diberikan masukan yang tidak sesuai dengan syntax, interpreter sudah dapat memberikan notifikasi kesalahan. Namun notifikasi tersebut belum mampu memberi informasi lokasi dimana kesalahan tersebut terjadi.
2. Penggunaan interpreter memudahkan pengaturan gerakan robot NAO bagi anak kecil usia SD dan SMP karena tidak perlu menentukan jalur interpolasi yang akan dilakukan oleh setiap motor, melainkan hanya perlu memasukan sebuah perintah untuk pengaturan gerakan robot NAO.
3. Robot NAO kehilangan keseimbangan ketika melakukan gerakan taichi, gerakan berdiri, dan gerakan duduk dengan parameter kecepatan tujuh.
4. Robot NAO kehilangan keseimbangan pada kombinasi gerakan condong kebelakang + angkat lengan kanan/kiri keatas 105 derajat.
5. Dari 10 anak yang mencoba program Interpreter INI, semuanya mampu membuat gerakan untuk robot NAO.

### 5.2 Saran

Untuk pengembangan lebih lanjut mengenai tugas akhir ini, disarankan untuk melakukan beberapa langkah lanjutan:

1. Penambahan daftar gerakan baik untuk gerakan tanpa parameter dan gerakan dengan parameter.
2. Diperlukan mekanisme untuk mendeteksi lokasi dimana kesalahan terjadi pada masukan.
3. Penambahan modul Naoqi yang digunakan oleh eksekutor pada interpreter terutama modul sensor yang memiliki potensi

untuk membuat program menjadi lebih baik.

4. Keluaran konsol ketika terjadi kesalahan masukan dapat dibuat interaktif berupa pertanyaan kepada pengguna sehingga lebih menarik.

## DAFTAR PUSTAKA

- [1] H. D. Siswaja, Prinsip kerja dan klasifikasi robot. Media Informatika, 2008. (Dikutip pada halaman ix, 5, 6).
- [2] M. Chang Wu, The Development of Screenplay Interpreter for Multi-Morphic Robots. PhD thesis, Department of Electrical Engineering National Sun Yat-sen, Aug. 2012. (Dikutip pada halaman ix, 7, 8).
- [3] A. Robotics, User Guide version 1.8.11. francis: NAO Aldebaran Robotics, 2008. (Dikutip pada halaman ix, xi, 8, 10, 11, 12, 13, 14, 15, 16, 31, 32, 36).
- [4] I. Icemanind, “Easily create your own parser - CodeProject.” <http://www.codeproject.com/Articles/220042/Easily-Create-Your-Own-Parser>. [Online; diakses 2-Desember-2014]. (Dikutip pada halaman ix, 20, 21).
- [5] “Introduction to robots.” <http://www.galileo.org/robotics/intro.html>. [Online; diakses 12-Desember-2014]. (Dikutip pada halaman 5).
- [6] B. Choi, Humanoid Robotic Language and Virtual Reality Simulation. Rijeka: In- Tech, humanoid robot ed., 2009. (Dikutip pada halaman 7).
- [7] W. M. Hinojosa, N. G. Tsagarakis, and D. G. Caldwell, Performance Assessment of a 3 DOF Differential Based Waist joint for the iCub Baby Humanoid Robot. Rijeka: In- Tech, humanoid robot ed., 2009. (Dikutip pada halaman 7).
- [8] C. Zhou, “ISYE 4256 SUPPLEMENTAL MATERIAL,” School of Industrial and Systems Engineering Georgia Institute of Technology, 1999. (Dikutip pada halaman 7).
- [9] E. Berger, Master Thesis Berger. Freiberg: Technical University Bergakademie Freiberg, 2011. (Dikutip pada halaman 10, 14).
- [10] A. Muliantara, PENERAPAN REGULAR EXPRESSION DALAM MELINDUNGI ALAMAT EMAIL DARI SPAM

ROBOT PADA KONTEN WORDPRESS. PhD thesis, Program Studi Teknik Informatika, Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Udayana, 2009. (Dikutip pada halaman 15).

- [11] C. Asnawi, "Compiler dan interpreter." <http://asnawi.stmikayani.ac.id/materi/pdp/ModulPDPBab3.pdf>. [Online; diakses 13-Desember-2014]. (Dikutip pada halaman 17).
- [12] "Interpreter (computing)." [http://en.wikipedia.org/wiki/Interpreter\\_computing](http://en.wikipedia.org/wiki/Interpreter_computing). (Dikutip pada halaman 17).
- [13] J. Conrod, "A simple interpreter from scratch in python." <http://jayconrod.com/posts/37/a-simple-interpreter-from-scratch-in-python-part-1>. [Online; diakses 12-Desember-2014]. (Dikutip pada halaman ix, 19).
- [14] "The NAO-base // f.u.n. lab // university of notre dame." <http://funlab.nd.edu/the-nao-base/>. [Online; diakses 10-Desember-2014]. (Dikutip pada halaman 29).
- [15] T. Pavel, "Autocomplete menu - CodeProject." <http://www.codeproject.com/Articles/365974/Autocomplete-Menu>. [Online; diakses 2-Desember-2014]. (Dikutip pada halaman 23).

## BIOGRAFI PENULIS



**Eka Prasetyo Herwidodo**, lahir pada 19 oktober 1992 di Kota Madiun, Jawa Timur. Penulis lulus dari SMP Negeri 2 Madiun pada tahun 2007 kemudian melanjutkan pendidikan ke SMA Negeri 2 Madiun hingga akhirnya lulus pada tahun 2010. Penulis kemudian melanjutkan pendidikan Strata satu ke Jurusan Teknik Elektro ITS Surabaya bidang studi Teknik Komputer dan Telematika. Saat di bangku kuliah penulis aktif menjadi anggota kalam kepengurusan 2011/2012 dan menjadi ketua departemen Media pada kepengurusan 2012/2013. Sejalan dengan itu, penulis juga aktif menjadi Asisten laboratorium B201 Telematika dan kemudian B401 Informatika digital hingga saat ini. Belakangan, penulis tertarik dengan riset mengenai robotika terutama humanoid robot.